A Parameterised Complexity Analysis of Bi-level Optimisation with Evolutionary Algorithms

Dogan Corus psxdc1@nottingham.ac.uk
School of Computer Science, University of Nottingham, Nottingham, NG8 1BB, UK

Per Kristian Lehre perkristian.lehre@nottingham.ac.uk
School of Computer Science, University of Nottingham, Nottingham, NG8 1BB, UK

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

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

doi:10.1162/EVCO_a_00147

Abstract

Bi-level optimisation problems have gained increasing interest in the field of combinatorial optimisation in recent years. In this paper, we analyse the runtime of some evolutionary algorithms for bi-level optimisation problems. We examine two NP-hard problems, the generalised minimum spanning tree problem and the generalised travelling salesperson problem in the context of parameterised complexity. For the generalised minimum spanning tree problem, we analyse the two approaches presented by Hu and Raidl (2012) with respect to the number of clusters that distinguish each other by the chosen representation of possible solutions. Our results show that a (1+1) evolutionary algorithm working with the spanning nodes representation is not a fixed-parameter evolutionary algorithm for the problem, whereas the problem can be solved in fixed-parameter time with the global structure representation. We present hard instances for each approach and show that the two approaches are highly complementary by proving that they solve each other's hard instances very efficiently. For the generalised travelling salesperson problem, we analyse the problem with respect to the number of clusters in the problem instance. Our results show that a (1+1) evolutionary algorithm working with the global structure representation is a fixed-parameter evolutionary algorithm for the problem.

Keywords

Bi-level optimisation, evolutionary algorithms, combinatorial optimisation.

1 Introduction

Many interesting combinatorial optimisation problems are hard to solve, and metaheuristic approaches such as local search, simulated annealing, evolutionary algorithms (EAs), and ant colony optimisation have been used for a wide range of these problems.

In recent years, researchers have become very interested in bi-level optimisation for single-objective (Koh, 2007; Legillon et al., 2012) and multiobjective (Deb and Sinha, 2009, 2010) problems. Such problems can be split into an upper-level and a lower-level problem, which depend on each other. By fixing a possible solution for the

Manuscript received: December 2, 2013; revised: July 29, 2014; accepted: January 7, 2015.

upper-level problem, the lower-level problem is optimised with respect to the given objective and the constraints imposed by the choice of the upper-level problem.

Sinha et al. (2014) give the following general definition of a bi-level optimisation problem.

DEFINITION 1 (SINHA ET AL., 2014). *Let $X = X_U \times X_L$ denote the product of the upper-level decision space X_U and the lower-level decision space X_L, i.e., $x = (x_u, x_\ell) \in X$, if $x_u \in X_U$ and $x_\ell \in X_L$. For an upper-level objective function $F : X \to \mathbb{R}$ and a lower-level objective function $f : X \to \mathbb{R}$, a general bi-level optimisation problem is given by*

$$
\begin{aligned}
&Minimise_{x \in X} \quad F(x) \\
&\quad s.t. \\
&\qquad x_\ell \in arg \min_{x_\ell \in X_L} \{ f(x) \mid g_i(x) \geq 0, i \in I \}, \\
&\qquad G_j(x) \geq 0, \, j \in J,
\end{aligned}
$$

where the functions $g_i : X \to \mathbb{R}$, $i \in I$, represent lower-level constraints and $G_j : X \to \mathbb{R}$, $j \in J$, is the collection of upper-level constraints.

A vector $x = (x_u, x_l) \in X$ is called feasible on the upper level if it fulfils all upper-level constraints. For a given x_u the lower-level solution x_l has to be optimal. A comprehensive benchmark set in the context of continuous optimisation was introduced by Sinha et al. (2014).

According to Definition 1, the upper-level objective function F and the lower-level objective function f may constitute an arbitrary optimisation problem, which implies that they could both be difficult, that is, NP-hard problems. Bi-level problems can be found in a variety of domains such as transportation or economics. The toll-setting problem (Brotcorne et al., 2001) is one such problem from the transportation domain, in which the government that operates the highways tries to maximise its revenues by placing toll gates in a road network. Drivers have different objectives and avoid tolls by choosing paths that minimise their travelling costs. This example shows that the upper- and lower-level problems can work against each other, that is, the government tries to maximise revenue, whereas the drivers try to minimise their costs.

To initiate runtime analysis of evolutionary bi-level optimisation, we examine settings for the NP-hard generalised minimum spanning tree problem (GMSTP) where the upper- and lower-level problems are cooperative. In our case, the upper-level function constitutes an NP-hard optimisation problem, whereas the lower-level problem can be solved in polynomial time. We examine two approaches introduced by Hu and Raidl (2011, 2012) for the GMSTP. Both approaches work with an upper-layer and a lower-layer solution. The upper-layer solution x_u is evolved by an evolutionary algorithm, whereas the optimal solution x_l of the lower-layer problem corresponding to a particular search point x_u of the upper layer can be found in polynomial time using deterministic algorithms.

Our goal is to understand the two different approaches by parameterised computational complexity analysis (Downey and Fellows, 1999). The computational complexity analysis of metaheuristics plays a major role in the theoretical analysis of this type of algorithms and studies the runtime behaviour with respect to the size of the given input. We refer the reader to Auger and Doerr (2011) and Neumann and Witt (2010) for a comprehensive presentation. Parameterised complexity analysis takes into account the runtime of algorithms in dependence of an additional parameter that measures the

hardness of a given instance. This allows us to understand which parameters of a given NP-hard optimisation problem make it hard or easy to be optimised by heuristic search methods. In the context of evolutionary algorithms, the term fixed-parameter evolutionary algorithms has been defined by Kratsch and Neumann (2013). An evolutionary algorithm is called a fixed-parameter evolutionary algorithm for a given parameter k iff its expected runtime is bounded by $f(k) \cdot \text{poly}(n)$, where $f(k)$ is a function only depending on k, and $poly(n)$ is a polynomial with respect to the input size n. Parameterised computational complexity analysis of evolutionary algorithms has been carried out for the vertex cover problem (Kratsch and Neumann, 2013), the computation of maximum leaf spanning trees (Kratsch et al., 2010), makespan scheduling (Sutton and Neumann, 2012b), and the travelling salesperson problem (Sutton and Neumann, 2012a).

We push forward the parameterised analysis of evolutionary algorithms and present an analysis in the context of bi-level optimisation. In our investigations, we take into account two NP-hard problems: the GMSTP and the generalised travelling salesperson problem (GTSP), which share the parameter, number of clusters m. We consider two different bi-level representations for the GMSTP that both have a polynomially solvable lower-level part. For the *spanning nodes representation*, we present worst-case examples showing that there are instances leading to an optimisation time of $\Omega(n^m)$. For the *global structure representation*, we show that it leads to a fixed-parameter evolutionary algorithm with respect to the number of clusters m. Furthermore, we present an instance class where the algorithm using the global structure representation, encounters an optimisation time of $m^{\Omega(m)}$. Analysing both approaches on each other's worst-case instances, we show that they solve them very efficiently. This shows the complementary abilities of these two representations for the GMSTP. Then we extend our results for global structure representation to the GTSP to show that a similar algorithm has an expected optimisation time of $m^{\Omega(m)}$ for this problem as well.

The paper is divided into two main parts according to the two different problems. The first part, based on the conference version (Corus et al., 2013), where the GMSTP is investigated, is presented in Section 2. We show hard instances for the spanning nodes representation in Section 2.2 and show that a simple evolutionary algorithm needs exponential time even if the number of clusters is small. In Section 2.3 we examine the global structure representation and show that this leads to fixed-parameter evolutionary algorithms for the GMSTP. We point out complementary abilities in Section 2.4. This paper extends the conference version by investigations of the GTSP and some generalisations. We examine the GTSP with the corresponding global structure representation in Section 3 and provide upper and lower bounds on the optimisation time of the considered algorithm. In Section 4 we point out general characteristics that allow this fixed-parameter result to be extended to other problems.

2 Generalised Minimum Spanning Tree Problem

In this section, we consider the GMSTP and provide the runtime analysis with respect to bi-level representations given by Hu and Raidl (2011, 2012).

2.1 Preliminaries

We consider the GMSTP introduced by Myung et al. (1995). The input is given by an undirected complete graph $G = (V, E, c)$ on n nodes with a cost function $c : E \to \mathbb{R}^+$ that assigns positive costs to the edges. Furthermore, a partitioning of the node set V into m pairwise disjoint clusters $V_1, V_2, \ldots, V_m$ is given such that $n = \sum_{i=1}^{m} |V_i|$.

A solution to the GMSTP consists of two components, the m chosen nodes P, called *the spanning nodes*, in the m clusters, and a minimum spanning tree T on the graph induced by the spanned nodes. More precisely, a solution $S = (P, T)$ consists of a node set $P = (p_1, \ldots, p_m) \in V^m$, where $V^m = V_1 \times V_2 \times \cdots \times V_m$ and a spanning tree $T \subseteq E$ on the subgraph $G[P] = G(P, \{e \in E \mid e \subseteq P\})$ induced by P. The cost of T is the cost of the edges in T, namely,

$$C(T) = \sum_{(u,v)\in T} c(u, v).$$

The goal is to compute a solution $S^* = (P^*, T^*)$ that has minimal cost among all possible solutions $S = (P, T)$. For an easier presentation, we assume in some cases that edge costs can be ∞. In this case, we restrict our investigations to solutions that do not include edges with cost ∞. Alternatively, one might view this as the GMSTP defined on a graph that is not necessarily complete.

The GMSTP is NP-hard (Myung et al., 1995), and two different bi-level evolutionary approaches have been examined by Hu and Raidl (2012). The first approach they present uses the spanned nodes representation. It selects in the upper-level problem a node for each cluster and computes on the lower level a minimum spanning tree (using, for example, Kruskal's algorithm in time $O(m \log m)$) on the induced subgraph. Tabu search and variable neighbourhood approaches using this representation can be found in Ghosh (2003).

The second approach uses the global structure representation. It constructs a complete graph $H = (V', E')$ from the given input graph $G = (V, E, c)$ and the set of pairwise disjoint clusters $V_1, V_2, \ldots, V_m$. The node $v_i \in V'$, $1 \leq i \leq m$, corresponds to the cluster V_i in G. The search space for the upper level consists of all spanning trees of H, and the spanned nodes of the different clusters are selected in time $O(n^2)$ using the dynamic programming approach of Pop (2004).

For our theoretical investigations, we measure the runtime of the algorithms by the number of fitness evaluations required to obtain an optimal solution. We call this the *optimisation time* of the examined algorithm. The *expected optimisation time* refers to the expected number of fitness evaluations until an optimal solution has been obtained for the first time.

2.2 Spanned Nodes Representation

We analyse the cluster-based (1+1) EA in this section (see Algorithm 1). Our first theorem shows that this algorithm is an XP-algorithm (Downey and Fellows, 1999), that is, an algorithm that runs in time $O(n^{g(m)})$, where $g(m)$ is a computable function only depending on m, when choosing the number of clusters m as a parameter.

THEOREM 1: *For any instance of the GMSTP, the expected time until the cluster based (1+1) EA reaches the optimal solution is $O(n^m)$.*

PROOF: For any search point x, let $w(x) \in \{0, 1, \ldots, m\}$ denote the number of clusters where the spanned node representation includes a suboptimal node. If the algorithm chooses all $w(x)$ suboptimal clusters for mutation and selects the optimal node in each of them, then the optimal solution is obtained. Since $w(x) \leq m$, the probability that all suboptimal clusters are mutated in a single step is at least $(1/m)^m$. The probability of choosing the optimal node in cluster i is $1/|V_i|$. Thus, the probability of jumping to the

Algorithm 1 Cluster-based (1+1) EA

Choose a random spanned nodes set $P \in V^m$.
Let T be a MST of $G[P]$.
while termination condition not satisfied **do**
  $P' \leftarrow P$
  for $i \in [m]$ **do**
    with probability $1/m$, sample $p'_i \sim \text{Unif}(V_i)$.
  end for
  Let T' be the MST of $G[P']$.
  if $C(T') \leq C(T)$ **then**
    $P \leftarrow P'$
    $T \leftarrow T'$
  end if
end while

optimal solution from any search point is at least

$$(m)^{-m} \prod_{i=1}^{m} |V_i|^{-1}.$$

Since $\sum_{i=1}^{m} V_i = n$, it holds that

$$\prod_{i=1}^{m} \frac{1}{|V_i|} \geq (m/n)^m.$$

Therefore, the probability of reaching the optimal solution in one step is $\Omega(n^{-m})$, and the expected time to reach the optimal solution is bounded from above by $O(n^m)$. □

We now consider an instance of GMSTP that is difficult for the cluster-based (1+1) EA. The hard instance G_S for the spanning nodes representation is illustrated in Figure 1. It consists of m clusters, where one cluster is called the *central cluster*, and the $m-1$ other clusters are called *peripheral clusters*. Each cluster contains n/m nodes, and we assume that $n = m^2$ holds. The nodes in the peripheral clusters are called *peripheral nodes*, and the nodes in the central cluster are called *central nodes*. Within each cluster, one of the nodes is called *optimal* and is marked black in the figure. The remaining $(n/m) - 1$ nodes are called *suboptimal* nodes and are marked white in the figure. The instance is a bipartite graph, where edges connect peripheral nodes to central nodes. The cost of any edge between two optimal nodes is 1, the cost of any edge between two suboptimal nodes is 2. The cost of any edge between a suboptimal peripheral node and the optimal central node is n^2, and the cost of any edge between an optimal peripheral node and a suboptimal central node is n. A cluster is called *optimal* in a solution if the solution has chosen the optimal node in that cluster.

THEOREM 2: *Starting with an initial solution chosen uniformly at random, the expected optimisation time of the cluster-based (1+1) EA on G_S is $\Omega(n^m)$.*

Furthermore, for any constant $\epsilon > 0$, the probability of having obtained an optimal solution after at most $n^{(1-\epsilon)m}$ iterations is $e^{-\Omega(m)}$.

PROOF: We define two phases for the run of the (1+1) EA. The first phase consists of the first $n-1$ iterations, and the second phase starts at the end of the first phase and

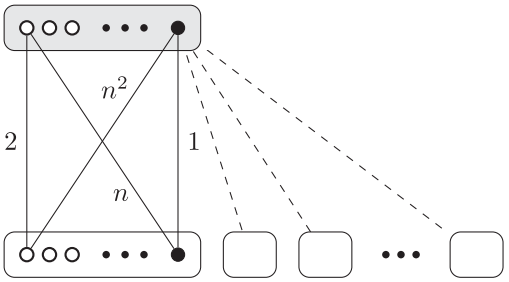

Figure 1: Hard instance G_S for spanning node representation.

continues for $n^{m/12}$ iterations. Four distinct events are considered failures during the run of the (1+1) EA for the instance just described:

- The first failure occurs if during the first phase of the run, the algorithm obtains a search point with less than $m/6$ suboptimal peripheral clusters.
- The second type of failure occurs when the central cluster fails to switch to a suboptimal node at least once during the first phase.
- The third type of failure corresponds to a direct jump to the optimal solution during the second phase.
- The fourth failure occurs when the algorithm does not switch all the optimal peripheral clusters to suboptimal clusters during the second phase.

We first show that the probability of the first failure event is at most $\exp(-\frac{m}{24})$. This implies that with overwhelmingly high probability, a constant fraction of peripheral clusters is always suboptimal during the first $n-1$ iterations. For $i \in [m-1]$ and $t \geq 0$, let $Z_i(t)$ be a random variable such that $Z_i(t) = 1$ if cluster V_i is always suboptimal in iteration 0 through iteration t, and $Z_i(t) = 0$ otherwise. The probability that a suboptimal node is selected in the initial solution is $1 - m/n$. In the following iterations, the probability that a cluster is selected for mutation and that its new spanned node is optimal is $(1/m)(m/n) = 1/n$. So it is clear that

$$\Pr(Z_i(t) = 1) \geq (1 - m/n)(1 - 1/n)^t.$$

By linearity of expectation,

$$\mathbf{E}\left[\sum_{i=1}^{m-1} Z_i(t)\right] \geq (m-1)\left(1 - \frac{m}{n}\right)\left(1 - \frac{1}{n}\right)^t.$$

Considering a phase length of $t = n - 1$, and assuming that m and n are sufficiently large and $n = m^2$ holds, we get

$$\mathbf{E}\left[\sum_{i=1}^{m-1} Z_i(t)\right] \geq \frac{m}{3}.$$

Finally, a Chernoff bound (Motwani and Raghavan, 1995) implies that

$$\Pr\left(\sum_{i=1}^{m-1} Z_i(t) \leq \left(1 - \frac{1}{2}\right)\frac{m}{3}\right) \leq \exp(-m/24).$$

We then show that the probability of the second failure event is $\exp(-\Omega(\sqrt{n}))$. In each iteration the probability to switch the central cluster to a suboptimal node is at least

$$p = \frac{1}{m}\left(1 - \frac{m}{n}\right) = \Omega\left(\frac{1}{\sqrt{n}}\right).$$

The probability that this event does not occur in $n-1$ steps is

$$\begin{aligned}(1-p)^{n-1} &= \left((1-p)^{1/p}\right)^{(n-1)p} \\ &\leq \exp(-p(n-1)) = \exp(-\Omega(\sqrt{n})).\end{aligned}$$

Now, we show that the probability of the third failure event is less than $n^{-m/12}$, assuming that the first two failure events do not occur. As long as the central cluster remains suboptimal, switching a suboptimal node in a peripheral cluster to an optimal node will result in an extra cost of $n-2$. Conversely, switching an optimal peripheral cluster into a suboptimal cluster will decrease the cost by $n-2$. As long as there is at least one suboptimal peripheral cluster, making the central cluster optimal will incur an extra cost of at least $n^2 - 2$. So, during phase two, the algorithm cannot make any suboptimal cluster optimal unless all suboptimal clusters are made optimal in the same iteration. The probability of making at least $m/6$ suboptimal peripheral clusters optimal simultaneously is at most

$$\left(\frac{1}{m} \cdot \frac{m}{n}\right)^{m/6} = \left(\frac{1}{n}\right)^{m/6}.$$

Since the probability to jump to the optimal solution is at most $n^{-m/6}$ in each iteration, it holds by the union bound that the probability of failure event three is at most

$$n^{-m/6} n^{m/12} = n^{-m/12}.$$

Finally, we show that the probability of failure event four is $O(n^{-m/13})$. For sufficiently large n, the probability that an optimal peripheral cluster is made suboptimal by the (1+1) EA is at least

$$\frac{1}{m} \cdot \frac{n-m}{n} \cdot \left(1 - \frac{1}{m}\right)^{m-1} \geq \frac{1}{3m}.$$

The expected time $E[T^{sub}]$ until all peripheral clusters have become suboptimal is therefore at most $m \cdot 3m = O(m^2)$. Considering a phase of length $n^{m/12}$ and taking into account $m^2 = n$, it holds by Markov's inequality that the probability of a type four failure is

$$\Pr\left(T^{sub} > n^{m/12}\right) \leq O(m^2) n^{-m/12} = O(n^{1-m/12}).$$

By union bound, the probability that any type of failure occurs is less than the sum of their independent probabilities, which is $e^{-\Omega(m)}$. Hence, with overwhelmingly high probability, after the second phase, the algorithm has obtained a locally optimal solution where all peripheral clusters are suboptimal. After that iteration, the probability to jump directly to the optimal solution is n^{-m}, and the expected time for this event to occur is n^m.

Let E be the event that no failure occurs. Then, the first statement of the theorem follows by the law of the total probability,

$$\begin{aligned}\mathbf{E}[T] &\geq \mathbf{E}[T|E]\Pr(E)\\ &= \Omega(n^m)(1-\mathrm{e}^{-\Omega(m)})\\ &= \Omega(n^m).\end{aligned}$$

Furthermore, by union bound, it holds that

$$\Pr\left(T < n^{(1-\epsilon)m} \mid E\right) \leq n^{(1-\epsilon)m}n^{-m} = n^{-\epsilon m}.$$

Hence, the second statement of the theorem follows by the law of total probability

$$\begin{aligned}\Pr\left(T < n^{(1-\epsilon)m}\right) &= \Pr\left(T < n^{(1-\epsilon)m} \mid E\right)\Pr(E)\\ &\quad + \Pr\left(T < n^{(1-\epsilon)m} \mid \overline{E}\right)\Pr\left(\overline{E}\right)\\ &\leq \Pr\left(T < n^{(1-\epsilon)m} \mid E\right) + \Pr\left(\overline{E}\right)\\ &\leq n^{-\epsilon m} + \mathrm{e}^{-\Omega(m)} = e^{-\Omega(m)}.\end{aligned}$$

□

Our results for the spanned nodes representation show that the cluster-based (1+1) EA obtains an optimal solution in time $O(n^m)$, and our analysis for the hard instance G_S shows that this bound is tight.

2.3 Global Structure Representation

The second approach examined by Hu and Raidl (2012) uses the global structure representation. It works on the complete graph $H = (V', E')$ obtained from the input graph $G = (V, E, c)$. The node $v_i \in V'$, $1 \leq i \leq m$, represents the cluster V_i of G.

The upper-level solution in the global structure representation is a spanning tree T of H, and the lower-level solution is a set of nodes $P = (p_1, \ldots, p_m)$, with $p_i \in V_i$ that minimises the cost of a spanning tree connecting the clusters in the same way as T. Given a spanning tree T of H, the set of nodes P can be computed in time $O(n^2)$ using dynamic programming (Pop, 2004).

We consider the tree-based (1+1) EA outlined in Algorithm 2. It starts with a spanning tree T of H that is chosen uniformly at random. In each iteration, a new solution T' of the upper layer is obtained by performing K edge swaps to T. Here the parameter K is chosen according to $1 + \mathrm{Pois}(1)$, where $\mathrm{Pois}(1)$ is the Poisson distribution with expectation 1. In one edge swap, an edge e currently not present in the solution is introduced and an edge from the resulting cycle is removed such that a new spanning tree of H is obtained. After having produced the offspring T', the corresponding set of nodes P' is computed using dynamic programming. P and T are replaced by P' and T' if the cost of the new solution is not worse than the cost of the old one.

In the following, we show that the tree-based (1+1) EA is a fixed-parameter evolutionary algorithm for the GMSTP when considering the number of clusters m as the parameter. We do this by transferring the result of Pop (2004) to the tree-based (1+1) EA.

THEOREM 3: *The expected time of the tree-based (1+1) EA to find the optimal solution for any instance of the GMSTP is $O(m^{3(m-1)})$. Furthermore, for any $k \geq 1$, the probability that an optimal solution is not found within $ekm^{3(m-1)}$ steps is less than $\exp(-k)$.*

Algorithm 2 Tree-based (1+1) EA

Choose a spanning tree T of H.
Apply dynamic programming to find the minimum spanned nodes $P = (p_1, \ldots, p_m)$ induced by T.
while termination condition not satisfied **do**
  $T' \leftarrow T$
  for $i \in [K]$ where $K \sim 1 + \text{Pois}(1)$ **do**
    Sample edge $e \sim \text{Unif}\,(E' \setminus T)$
    Sample edge $e' \sim \text{Unif}$(edges in cycle in $T' \cup \{e\}$)
    $T' \leftarrow T' \cup \{e\} \setminus \{e'\}$
  end for
  Apply dynamic programming to find a set of spanned nodes $P' = (p'_1, \ldots, p'_m)$ with respect to T' of minimal cost.
  if $\sum_{(i,j)\in T'} c(p'_i, p'_j) \leq \sum_{(i,j)\in T} c(p_i, p_j)$ **then**
    $P \leftarrow P'$
    $T \leftarrow T'$
  end if
end while

PROOF: An upper-layer solution is a tree T of H. Let T^* be any tree of H for which there exists a set P^* of spanning nodes such that T^* and P^* form an optimal solution. For any nonoptimal solution T, define $w(T)$ as the number of edges in T^* that are missing in T.

The mutation operator can convert a nonoptimal solution T into the optimal solution T^* with a sequence of $w(T) \leq m - 1$ edge exchange operations. The probability that the mutation operator exchanges $w(T) \leq m - 1$ edges in one mutation step is at least

$$\Pr(\text{Pois}(1) = m - 2) = 1/e(m-2)!.$$

In each exchange operation, if there are i optimal edges missing, then the probability that one of the missing optimal edges is inserted is at least i/m^2. After the addition of an optimal edge, the probability of excluding a nonoptimal edge is at least $1/m$, since the largest cycle cannot be longer than m. At most $m - 1$ nonoptimal edges must be exchanged in this manner. So the probability that the nonoptimal solution T will be converted to the optimal solution T^* in one mutation step is at least

$$\frac{1}{\mathrm{e}(m-2)!} \cdot \prod_{i=1}^{m-1} \frac{i}{m^2} \cdot \frac{1}{m} \geq (1/\mathrm{e})m^{-3(m-1)}.$$

So, the expected time to achieve an optimal solution is $O(m^{3(m-1)})$. Furthermore, the probability that the optimal solution has not been created after $ekm^{3(m-1)}$ iterations is

$$(1 - (1/\mathrm{e})m^{-3(m-1)})^{\mathrm{e}km^{3(m-1)}} \leq \exp(-k).$$

□

We now present an instance that is hard to be solved by the tree-based (1+1) EA. The instance G_G, illustrated in Figure 2, consists of n nodes and m clusters. There are two central clusters, denoted by V_1 and V_2. The cluster V_1 contains the two nodes v_{11} and v_{12}. The remaining clusters V_i, $2 \leq i \leq m$, contain a single node v_{i1} each. The edges that connect the nodes v_{11} to the peripheral cluster nodes have cost 1. The edges that

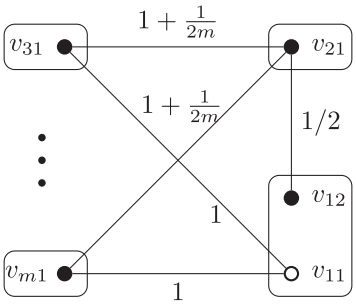

Figure 2: Hard instance G_G for global structure representation. Edges not shown have weight ∞.

connect v_{21} to the peripheral clusters have weight $1 + 1/2m$. The edge that connects v_{12} and v_{21} has weight $1/2$. All other edges have cost ∞. Hence, if the tree-based (1+1) EA connects cluster V_1 and V_2, then the dynamic programming algorithm will choose node v_{12}.

In our analysis, we will use the following lemma on basic properties of the Poisson distribution with expectation 1.

LEMMA 1: *If $K \sim Pois(1)$, then $Pr(K \geq n) < 2(e/n)^n$.*

PROOF: Using Stirling's approximation of the factorial,

$$n! > \sqrt{2\pi n}(n/\mathrm{e})^n > (n/\mathrm{e})^n,$$

we obtain the simple bound

$$\begin{aligned}
\Pr(K \geq n) &= \sum_{i=n}^{\infty} \frac{1}{\mathrm{e} i!} \\
&< \sum_{i=n}^{\infty} \frac{1}{i!} \\
&< \sum_{i=n}^{\infty} \frac{1}{n!} \left(\frac{1}{n+1}\right)^{i-n} \\
&< (\mathrm{e}/n)^n \sum_{i=0}^{\infty} \left(\frac{1}{n+1}\right)^{i} \\
&= (\mathrm{e}/n)^n \left(1 + \frac{1}{n}\right).
\end{aligned}$$

□

Using the previous lemma (Lemma 1), we are able to show that the tree-based (1+1) EA finds it hard to optimise G_G when the initial spanning tree is chosen uniformly at random among all spanning trees having weight less then ∞.

THEOREM 4: *Starting with a spanning tree chosen uniformly at random among all spanning trees that have cost less than ∞, the expected optimisation time of the tree-based (1+1) EA on G_G is $\Omega((m/e)^{m-1})$.*

PROOF: Consider the instance in Figure 2. In the following, edge $e^* := \{v_{12}, v_{21}\}$ is the edge connecting the two central clusters. The optimal solution corresponds to the spanning tree that includes edge e, and where all other clusters are connected to cluster V_2. The solution where all peripheral clusters are connected to V_1, and where cluster V_2 is connected to one of the peripheral clusters, is a local optimum.

We define four failure events that can occur during a run of the (1+1) EA on this instance:

- The first type of failure occurs when the initial solution includes edge e.
- The second type of failure occurs when less than $m/3$ of the peripheral clusters are connected to cluster V_1 in the initial solution.
- The third type of failure occurs when the algorithm jumps directly to the optimal solution during the first $((m-2)/3\mathrm{e})^{(m-2)/6}$ iterations.
- The fourth type of failure occurs if after iteration $((m-2)/3\mathrm{e})^{(m-2)/6}$, there exists a peripheral cluster that is not connected to cluster V_1.

There are $m-2$ peripheral clusters that must be connected to either V_1 or V_2. Additionally, cluster V_1 and V_2 must be connected. This connection can be established either by adding edge $e^* = (v_{12}, v_{21})$ or by connecting a peripheral cluster to both V_1 and V_2. There are 2^{m-2} spanning trees that contain edge e^*, and $(m-2) \cdot 2^{m-3}$ spanning trees that do not contain edge e^*, since one of the $m-2$ peripheral clusters will be connected to both central clusters and the others will be connected to only one. So, the probability that a uniformly chosen spanning tree includes edge e^* is $O(1/m)$, which is the probability of the first type of failure.

Now, we show that the probability of the second type of failure is at most $\exp(-\Omega(m))$. Considering that the probability of a specific cluster is adjacent to V_1 in the initial solution is at least $1/2$, the probability that less than $(m-2)/3$ clusters are connected to cluster V_1 in the initial solution is bounded by $\exp(-\Omega(m))$ using a Chernoff bound.

Assuming that type one and type two failures did not occur, the algorithm cannot accept new search points where a cluster originally connected to V_1 is instead connected to V_2, since it will create an extra cost of $1/2m$. The only exception is if a type three failure occurs, that is, the algorithm jumps directly to the optimal solution where all the peripheral clusters are connected to V_2. For a type three failure to occur, at least $(m-2)/3$ clusters have to be modified simultaneously. Therefore, using Lemma 1, the probability of jumping directly to the optimal solution in a single step is bounded from above by

$$2(3\mathrm{e}/(m-2))^{(m-2)/3}.$$

Taking a phase length of $((m-2)/3\mathrm{e})^{(m-2)/6}$ into account, the probability of a type three failure can be bounded from above using the union bound, as

$$((m-2)/3\mathrm{e})^{(m-2)/6} 2(3\mathrm{e}/(m-2))^{(m-2)/3} = (m/\mathrm{e})^{-\Omega(m)}.$$

Now, we show that the probability of a type four failure is $\mathrm{e}^{-\Omega(m)}$. The probability that a single peripheral cluster connected to V_2 is switched to V_1 is bounded from below by

$$\tfrac{1}{3} \cdot \tfrac{1}{\mathrm{e}} \cdot \tfrac{1}{(m^2-(m-1))} = \Omega(1/m^2).$$

Thus, the expected time between any such event is $O(m^2)$, and the expected time until all of the at most $m-2$ peripheral clusters are connected to V_1 is $\mathbf{E}[T'] = O(m^3)$. By Markov's inequality, it holds for any non-negative random variable X that

$$\Pr(X \geq k) \leq \frac{\mathbf{E}[X]}{k}.$$

The probability that it takes longer than

$$k = ((m-2)/3\mathrm{e})^{(m-2)/6}$$

iterations is therefore no more than

$$\frac{\mathbf{E}[T']}{k} = O(m^3) \cdot ((m-2)/3\mathrm{e})^{-(m-2)/6} = (m/\mathrm{e})^{-\Omega(m)}.$$

This proves our claim about the probability of failure event four.

If none of the above-mentioned failures occur, we reach the local optimum where all the peripheral clusters are connected to cluster V_1. From this point on, the probability to jump to the optimal solution is by Lemma 1 no more than

$$2(\mathrm{e}/(m-1))^{m-1}$$

because it is necessary to make at least $m-1$ edge exchanges to reach the optimum. The expected time to reach the optimal solution conditional on no failure is therefore more than $(1/2)(m/e)^{m-1}$.

Let R be the event that no failure occurs. By the law of total probability, it follows that the expected time $\mathbf{E}[T]$ to reach the global optimum is

$$\begin{aligned}\mathbf{E}[T] &\geq \mathbf{E}[T|R]\Pr(R)\\ &= \Omega((m/\mathrm{e})^{m-1})(1-O(1/m))\\ &= \Omega((m/\mathrm{e})^{m-1}).\end{aligned}$$

□

The previous theorem shows that there are instances for the cluster-based (1+1) EA where the optimisation time grows exponentially with the number of clusters. In the next section, we compare the two different representations for GMSTP and show that they have complementary capabilities.

2.4 Complementary Capabilities

The two representations examined in the previous sections differ significantly from each other. They both rely on the fact that there is a deterministic algorithm that solves the lower-level problem in polynomial time. In this section, we want to examine the differences between the two approaches. We show that both representations have complementary abilities and do this by examining the algorithms on each other's hard instance. Surprisingly, we find that the hard instance for one algorithm becomes easy to solve when giving it as an input to the other algorithm.

In Section 2.2 we showed a lower bound of $\Omega(n^m)$ for the cluster-based (1+1) EA using the spanning node representation. The hard instance G_S for the cluster-based (1+1) EA given in Figure 1 consists of a central cluster to which all the other clusters are connected. There are no other connections between the clusters. Hence, there is only one spanning tree when working with the global structure representation. The dynamic programming algorithm that runs on the lower layer of the tree-based (1+1) EA therefore solves the problem in its first iteration.

The following theorem shows that these instances are easy to be optimised by the tree-based (1+1) EA.

THEOREM 5: *The tree-based (1+1) EA solves the instance G_S in expected constant time.*

PROOF: There is only a single tree over the cluster graph. Hence, the algorithm selects the optimal tree in the initial iteration. □

For the tree-based (1+1) EA, working with the global structure representation, we showed that it finds the instance G_G given in Figure 2 hard to solve. Working with the spanning nodes representation, there is only one cluster that consists of two nodes where all the other clusters contain exactly one node. Hence, an optimal solution is obtained by computing a minimum spanning tree on the lower level if the right node in the cluster of two nodes is chosen. The following theorem summarises this and shows that this instance becomes easy when working with the cluster-based (1+1) EA.

THEOREM 6: *The cluster-based (1+1) EA solves the instance G_G in expected time $O(m)$.*

PROOF: Cluster V_1 contains two nodes, and all other clusters contain a single node. If the initial solution is not already the optimal solution, the correct node of V_1 has to be selected using mutation. The node for the cluster V_1 is changed with probability $1/m$, and in such a step the correct node is selected with probability 1/2. Hence, the probability of a mutation leading to an optimal solution is at least $\frac{1}{2m}$ and the expected waiting time for this event is $O(m)$. □

The investigations show that the two examined representations have complementary abilities. Switching from one representation to the other one can significantly reduce the runtime.

3 Generalised Travelling Salesperson Problem

We now turn our attention to the NP-hard generalised travelling salesperson problem. Given a complete graph $G = (V, E, c)$ with a cost function $c : E \to \mathbb{R}^+$ and a partitioning of the node set V into m clusters V_i, $1 \leq i \leq m$, the goal is to find a cycle of minimal cost that contains exactly one node from each cluster.

The bi-level approach that we are studying is similar to the one discussed in the previous section. We investigate the global structure representation, which works on the complete graph $H = (V', E')$ obtained from the input graph $G = (V, E, c)$. The node $v_i \in V'$, $1 \leq i \leq m$, represents the cluster V_i of G.

The upper-level solution in the global structure representation is a Hamiltonian tour π on H, and the lower-level solution is a set of nodes $P = (p_1, \ldots, p_m)$ with $p_i \in V_i$ that minimises the cost of a Hamiltonian tour, which connects the clusters in the same way as π. Given the restriction imposed by the Hamiltonian tour π of H, finding the optimal set of nodes P can be done in time $O(n^3)$ by using any shortest path algorithm. One such algorithm is *cluster optimisation*, proposed initially by Fischetti et al. (1997) and widely used in the literature. Let $\pi = (\pi_1, \ldots, \pi_m)$ be a permutation on the m clusters and p_i be the chosen node for cluster V_{π_i}, $1 \leq i \leq m$. Then the cost of the tour π is

$$c(\pi) = c(p_m, p_1) + \sum_{i=1}^{m-1} c(p_i, p_{i+1}).$$

Our proposed algorithm (Algorithm 3) starts with a random permutation of clusters, which is always a Hamiltonian tour π, in a complete graph H. In each iteration, a new

Algorithm 3 Tour-based (1+1) EA

Choose a random permutation π (which is also a Hamiltonian tour) of the m given clusters.
Find the set of nodes P (one node in each cluster) to build the shortest path possible among those clusters with the given order, by means of any shortest path algorithm in time $O(n^3)$.
while termination condition not satisfied **do**
  $\pi' \leftarrow \pi$
  for $i \in [K]$ where $K \sim 1 + \text{Pois}(1)$ **do**
    Choose a node and a position in π' uniformly at random.
    $\pi' \leftarrow$ Perform the *Jump* with the chosen nodes on π'
  end for
  Find the set of nodes $P' = (p'_1, \ldots, p'_m)$ which minimises the cost with respect to π' in the lower level
  if $c(\pi') \leq c(\pi)$ **then**
    $P \leftarrow P'$
    $\pi \leftarrow \pi'$
  end if
end while

solution π' of the upper layer is obtained by the commonly used *jump* operator, which picks a node and moves it to a random position in the permutation. The number of jump operations carried out in a mutation step is chosen according to $1 + \text{Pois}(1)$, where Pois(1) denotes the Poisson distribution with expectation 1. Although we are using the jump operator in these investigations, similar results can be obtained for other popular mutation operators such as *exchange* and *inversion*.

THEOREM 7: *The expected optimisation time of the tour-based (1+1) EA is* $O(m!m^{2m})$.

PROOF: We consider the probability of obtaining the optimal tour π^* on the global graph H from an arbitrary tour π. The number of jump operations required is at most m (the number of clusters). The probability of picking the right node and moving it to the right position in each of those m operations is at least $1/m^2$. We can obtain an optimal solution by carrying out a sequence of m jump operations where the ith operation jumps element π^*_i in π to position i. Since the probability of $\text{Pois}(1) + 1 = m$ is $1/(\text{e}(m-1)!)$, the probability of a specific sequence of m jump operations to occur is bounded below by

$$\frac{1}{\text{e}(m-1)!} \cdot \frac{1}{m^{2m}}.$$

Therefore, the expected waiting time for such a mutation is

$$\left(\frac{1}{\text{e}(m-1)!} \cdot \frac{1}{m^{2m}}\right)^{-1} = O(m!m^{2m}),$$

which proves the upper bound on the expected optimisation time. □

Note that this upper bound depends on the number of clusters. Since the computational effort required to assess the lower-level problem is polynomial in input size, $O(n^3)$, this implies that the proposed algorithm is a fixed-parameter evolutionary algorithm for the GTSP problem and the parameter m, the number of clusters.

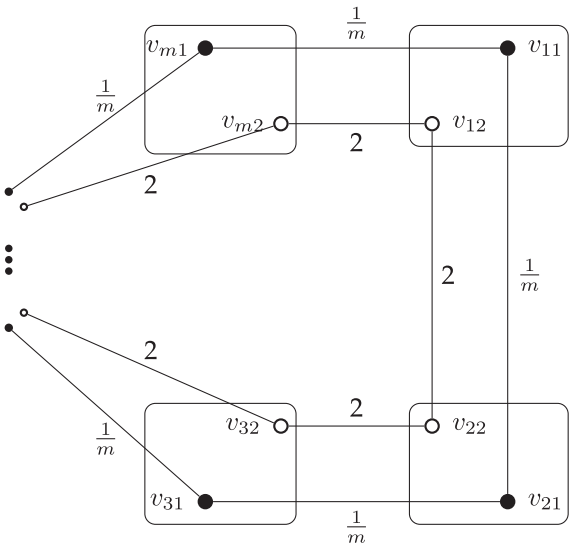

Figure 3: Hard instance G_G for GTSP with global structure representation.

So far we have found an upper bound for the expected time of finding an optimal solution using the presented algorithm. Next, we find a lower bound for the optimisation time. Figure 3 illustrates an instance G_G of GTSP for which finding the optimal solution is difficult by means of the presented bi-level evolutionary algorithm with global structure representation. In this graph, each cluster has two nodes. On the upper layer a tour for clusters is found by the EA, and on the lower layer the best node for that tour is found within each cluster. All white nodes (which represent suboptimal nodes) are connected to each other, making any permutation of clusters a Hamiltonian tour even if the black nodes are not used. All such connections have a weight of 1, except for those that are shown in the figure and have a weight of 2. All edges between a black node and a white node and also all edges between black nodes have weight m^2, except the ones presented in the figure that have weight $1/m$. An optimal solution of cost 1 uses only edges of cost $1/m$ whereas local optimal solutions use only edges of cost 1. The tour comprising all black nodes in the same order as illustrated in Figure 3 is the optimal solution. Note that there are many local optimal solutions of cost m. For our analysis it is just important that they do not share any edge with the global optimal solution.

The clusters are numbered in the figure, and a measure S for evaluating cluster orders is based on this numbering: Let $\pi = (\pi_1, \ldots, \pi_m)$ represent the permutation of clusters in the upper layer; then $S(\pi) = |\{i \,|\pi_{(i+1 \bmod m)} = (\pi_i + 1) \bmod m\}|$ indicates the similarity of the permutation with the optimal permutation. A large value of $S(\pi)$ means that many clusters in π are in the same order as in the optimal solution. Note that $S(\pi^*) = m$ for an optimal solution π^*. A solution π with $S(\pi) = 0$ is locally optimal in the sense that there is no strictly better solution in the neighbourhood induced by the jump operator. The solutions with $S(\pi) = 0$ form a plateau where all solutions differ from the optimal solution by m edges.

We first introduce a lemma that will later help us with the proof of the lower bound on the optimisation time.

LEMMA 2: *Let π and π' be two nonoptimal cluster permutations for the instance G_G. If $S(\pi') > S(\pi)$, then $c(\pi') > c(\pi)$.*

PROOF: In the given instance, all white nodes are connected to each other with a maximum weight of 2. These connections ensure that any permutation of the clusters can result in a Hamiltonian tour with a cost of at most $2m$. Moreover, all connections between white nodes and black nodes have a weight of m^2. So the lower level will never choose a combination of white and black nodes because the cost will be more than m^2, while there is an option of selecting all white nodes with the cost of at most $2m$. On the other hand, for any permutation of clusters other than the global optimum, the lower level will not choose any black nodes because it will not be possible to use all the $1/m$ edges and some m^2-weighted edges will be used again. Let $a = S(\pi)$ be the number of clusters adjacent to each other correctly from the right side (having the same rightside neighbour as in the global optimum) in a solution π. Then $b = m - a$ of clusters that have a different neighbour on their right. If π is not the optimal solution, then the lower level will choose all white nodes. As a result, a edges with weight 2 and b edges with weight 1 will be used in that solution; therefore, the total cost of solution π will be $c(\pi) = 2a + b = 2a + m - a = m + a$. Consider a solution π' with $a' = S(\pi')$ and $S(\pi') > S(\pi)$. We have $c(\pi') = m + a' > m + a = c(\pi)$, which completes the proof. □

Lemma 2 shows that any nonoptimal offspring π' of a solution π is not accepted if it is closer to an optimal solution π^*. This means that the algorithm finds it hard to obtain an optimal solution for G_G and leads to an exponential lower bound on the optimisation time, as shown in the following theorem.

THEOREM 8: *Starting with a permutation of clusters chosen uniformly at random, the optimisation time of the tour-based (1+1) EA on G_G is $(\frac{m}{2})^{\frac{2m}{3}}$ with probability $1 - e^{-\Omega(m)}$.*

PROOF: Considering G_G illustrated in Figure 3, the optimal solution is the tour comprising all edges with weight $\frac{1}{m}$. We consider a typical run of the algorithm consisting of a phase of $T = Cm^{3+\delta}$ steps, where C is an appropriate constant. For the typical run we show the following:

- A local optimum π with $S(\pi) = 0$ is reached with probability $1 - e^{-\Omega(m)}$.
- The global optimal solution is not obtained with probability $1 - m^{-\Omega(m)}$.

Then we state that only a direct jump from the local optimum to the global optimum is possible, and the probability of this event is $O(m^{-m/2})$.

First we show that with high probability $S(\pi_{init}) \leq \varepsilon m$ holds for the initial solution π_{init}, where ε is a small positive constant.

We count the number of permutations in which at least $\varepsilon m, \varepsilon > 0$ a small constant, of cluster neighbourhoods are correct. We should select εm of the clusters to be followed by their specific neighbour and consider the number of different permutations of $m - \varepsilon m$ clusters:

$$\binom{m}{\varepsilon m}(m - \varepsilon m)! \tag{1}$$

Some solutions are double-counted in this expression, so the actual number of different solutions with $S(\pi) \geq \varepsilon m$ is less than (1). Therefore, the probability of having

more than εm clusters followed by their specific cluster, is at most

$$\binom{m}{\varepsilon m}\frac{(m-\varepsilon m)!}{m!} = ((\varepsilon m)!)^{-1} = O\left(\left(\frac{\varepsilon m}{2}\right)^{-\frac{\varepsilon m}{2}}\right).$$

Hence, with probability $1 - O((\frac{\varepsilon m}{2})^{-\frac{\varepsilon m}{2}})$, $S(\pi_{init}) \leq \varepsilon m$ holds, and the initial solution has at most εm correctly ordered clusters.

Now we analyse the expected time to reach a solution π with $S(\pi) = 0$. For this purpose, we first consider the exchange operation and find the minimum number of different exchanges at each step that reduce the number of good orderings in a solution.

If we show the permutation of clusters for the current solution by $\pi = (\pi_1, ..., \pi_m)$, then there are $l = S(\pi)$ clusters in this permutation that are followed by their consecutive cluster. Note that for any solution other than the local optimum, $l > 0$ holds. Let j be one of these clusters, which is followed by cluster $j+1$. In order to destroy this good ordering, cluster j should be exchanged with a cluster r that fulfils the following requirements:

- Cluster r cannot be a consecutive cluster of j's current neighbours, that is, if we name j's neighbours i and k, then r cannot be $i-1, i+1, k-1, k+1$ because these nodes will introduce a new good ordering to the solution if we replace j.
- The current position of cluster r' in the permutation should not be before or after clusters $j-1$ or $j+1$ because replacing r with j would introduce new good orderings in that case.

Therefore, the total number of positions in the permutation that should not be selected as r is at most 8, meaning that there are $m-8$ choices for r that result in reducing the number of good orderings. Since there are l choices for j and $m-8$ choices for r, the total number of possible exchange operations to reach a permutation π' with $S(\pi') < S(\pi)$ is

$$l \cdot (m-8).$$

On the other hand, it is possible to simulate each *exchange* operation with two jumps. For any j and r, *exchange*(j,r) can be implemented by performing *jump*(j,r) and *jump*(r, $j+1$). The first jump will place j before r, and the second one will place r before $j+1$. Now we find the probability that two jumps happen at one step and simulate one of the possible exchange operations as

$$P = l \cdot (m-8) \cdot \frac{1}{e} \cdot \frac{1}{m^2} \cdot \frac{1}{m^2}.$$

In this formula, $l \cdot (m-8)$ is the number of different choices for exchange operation, $\frac{1}{e}$ is the probability of performing two mutation operations at one step, and each $\frac{1}{m^2}$ is the probability of selecting the two specific nodes for a jump operation. Using this probability, the expected time until l decrease by one is

$$E(T) = \frac{1}{P} = \gamma \cdot \frac{m^3}{l},$$

where γ is an appropriate constant. The maximum value of l is $S(\pi_{init})$, which we have already proved is at most ϵm. Therefore, with summing up the expected time of reducing l gradually from its maximum value to 1, we can find the expected time to

reach the local optimum π with $S(\pi) = 0$ as

$$E(T_{LO}) = \sum_{l=\epsilon m}^{1} \gamma \cdot \frac{m^3}{l} = O(m^3 \log m).$$

If γ' denotes the appropriate constant that $E(T_{LO}) = \gamma' \cdot m^3 \log m$, then by Markov's inequality we have

$$Pr\left(T_{LO} > 2 \cdot \gamma' \cdot m^3 \log m\right) \leq \frac{1}{2}.$$

If we repeat phases of $2 \cdot \gamma' \cdot m^3 \log m$ iterations for $\frac{m^\delta}{\log m}$ times, the probability that the local optimum is not reached in any of them is at most

$$Pr\left(T_{LO} > 2 \cdot \gamma' \cdot m^3 \log m \cdot \frac{m^\delta}{\log m}\right) \leq \left(\frac{1}{2}\right)^{\frac{m^\delta}{\log m}} = e^{-\Omega(m^\delta)}.$$

As a result, with probability $1 - e^{-\Omega(m^\delta)}$ the algorithm will reach a local optimum in a phase of $2\gamma' \cdot m^{3+\delta}$ steps, which if we consider $C = 2\gamma'$, is actually the same as the phase of $T = C \cdot m^{3+\delta}$ iterations that we mentioned previously.

To prove that with high probability the global optimum is not reached during the considered phase, first note that by Lemma 2 any jump to a solution closer to the optimum other than directly to the global optimum will be rejected. Furthermore, for the initial solution $S(\pi_{init}) \leq \varepsilon m$. Therefore, only nonoptimal solutions π with $S(\pi) \leq \varepsilon m$ are accepted by the algorithm. In order to obtain an optimal solution the algorithm has to produce the optimal solution from a solution π with $S(\pi) \leq \varepsilon m$ in a single mutation step. We now upper-bound the probability of such a direct jump that changes at least $(1 - \varepsilon)m$ clusters to their correct order. Such a move needs at least $\frac{(1-\varepsilon)m}{3}$ operations in the same iteration because each jump can change at most three edges. Taking into account that these jump operations may be acceptable in any order, the probability of a direct jump is at most

$$\frac{1}{e\left(\frac{(1-\varepsilon)m}{3}\right)!} \cdot \frac{1}{m^{\frac{2(1-\varepsilon)m}{3}}} \cdot \left(\frac{(1-\varepsilon)m}{3}\right)! = m^{-\Omega(m)}. \tag{2}$$

So in a phase of $O(m^{3+\delta})$ iterations the probability of having such a direct jump is by union bound at most $m^{-\Omega(m)+3+\delta} = m^{-\Omega(m)}$.

So far we have shown that a local optimum π with $S(\pi) = 0$ is reached with probability $1 - \mathrm{e}^{-\Omega(m^\delta)}$ within the first $T = Cm^{3+\delta}$ iterations. The probability of obtaining an optimal solution from a solution π with $S(\pi) = 0$ is at most

$$\frac{1}{\mathrm{e}\left(\frac{m}{3}\right)!} \cdot \frac{1}{m^{\frac{2m}{3}}} \cdot \left(\frac{m}{3}\right)! = \mathrm{e}^{-1} \cdot m^{-\frac{2m}{3}}.$$

We now consider an additional phase of $(\frac{m}{2})^{\frac{2m}{3}}$ steps after having obtained a local optimum. Using the union bound, the probability of reaching the global optimum in this phase is at most

$$\left(\frac{m}{2}\right)^{\frac{2m}{3}} \cdot \mathrm{e}^{-1} \cdot m^{-\frac{2m}{3}} \leq \left(\frac{1}{2}\right)^{\frac{2m}{3}}.$$

As a result, the probability of not reaching the optimal solution in these $(\frac{m}{2})^{\frac{2m}{3}}$ iterations is $1 - 2^{-\frac{2m}{3}} = 1 - \mathrm{e}^{-\Omega(m)}$. Altogether, the optimisation time is at least $(\frac{m}{2})^{\frac{2m}{3}}$ with probability $1 - \mathrm{e}^{-\Omega(m)}$. □

4 Discussion of Generalisations

The problems we have examined in this work are bi-level optimisation problems where the upper-level problem, namely the *leader*, and the lower-level problem, the *follower*, share an objective function. The general bi-level optimisation problem also includes the setting where the leader and the follower have different objectives. Given the decision of the leader, the follower makes a decision according to its objective function, which might conflict with the objective function of the leader. An example of such a problem is where the leader places toll booths across a road network and the followers try to find the cheapest way from a point A to a point B by finding a path that avoids as many toll booths as possible. Here, the leader can only learn the objective function value of its decision after the follower picks the optimum path. Unlike the GMSTP and GTSP, the objective functions of upper- and lower-level problems are distinct and conflicting in this toll booth problem.

For a given solution visited in the upper-level problem, the evaluation cost is, in the worst case, the computational complexity of the lower-level problem. If the lower-level problem can be solved in polynomial time, then a fixed-parameter bound on the size of the upper-level solution is sufficient for the overall problem to be fixed-parameter tractable. Because when the size of the upper-level solution is bounded by a function $f(k)$ that only depends on a parameter k of the original problem, the search process in the upper level will occur in a space of size $2^{f(k)}$, and a simple heuristic like the uniform random search will be able to find the optimal upper-level solution in $2^{f(k)}$ iterations and $2^{f(k)} \cdot poly(n)$ basic operations in expectation.

In our case, the global structure representation of GMSTP and GTSP, the size of an upper-level solution is bounded above by m^2, since it is enough to indicate with a single bit whether any two clusters are connected to precisely define a solution. On the other hand, the spanned nodes representation of GMSTP needs an upper-level solution to indicate *which node* is selected in *each cluster*. The required number of bits in the upper-level solution is maximised when there are $\frac{n}{m}$ nodes in each cluster because of inequality of arithmetic and geometric mean, and the maximum number of bits required is $m \log(\frac{n}{m})$. With global structure representation, if we pick our solutions uniformly at random, the probability of picking a unique optimal solution is $(1/2)^{m^2}$, which will occur in $O(2^{m^2})$ trials in expectation. Therefore uniform random search is fixed-parameter tractable for the problem. However with the spanned node representation the corresponding upper bound is $O(2^{m \log(\frac{n}{m})}) = O((n/m)^m)$, which still depends on the solution size n and does not provide information about the fixed-parameter tractability of the problem. However, it is noteworthy that the upper bound with the uniform random search is asymptotically better than the lower bound $\Omega(n^m)$ provided for the hard instance presented for the proposed algorithms. Nevertheless, the performance of the random uniform search ($O((n/m)^m)$) would still fall into the category of XP-algorithms, in accordance with the FPT-XP distinction between the representations.

5 Conclusions

Evolutionary bi-level optimisation has attracted increasing interest in recent years. With this article we have contributed to the theoretical understanding by considering two classical NP-hard combinatorial optimisation problems, namely, the generalised minimum spanning tree and the generalised travelling salesperson. We studied evolutionary algorithms for these problems in the parameterised setting. Using parameterised computational complexity analysis of evolutionary algorithms for the GMSTP, we examined

two representations for the upper-layer solutions and their corresponding deterministic algorithms for the lower layer. Our results show that the global structure representation leads to fixed-parameter evolutionary algorithms. By presenting hard instances for each of the two approaches, we pointed out where they run into difficulties. Furthermore, we showed that the two representations for the GMSTP are highly complementary by proving that they are highly efficient on the hard instance of the other algorithm. After having achieved these results for the GMSTP, we turned our attention to the GTSP. We showed that using the global structure representation leads to fixed-parameter evolutionary algorithms with respect to the number of clusters. Furthermore, we pointed out a worst-case instance where the optimisation time grows exponential with respect to the number of clusters, and we discussed generalisations of the results.

Acknowledgments

The research leading to these results received funding from the Australian Research Council (ARC) under grant agreements DP130104395 and DP140103400, and from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 618091 (SAGE).

References

Auger, A., and Doerr, B. (Eds.) (2011). *Theory of randomized search heuristics: Foundations and recent developments*. Singapore: World Scientific.

Brotcorne, L., Labbé, M., Marcotte, P., and Savard, G. (2001). A bilevel model for toll optimization on a multicommodity transportation network. *Transportation Science*, 35(4): 345–358.

Corus, D., Lehre, P. K., and Neumann, F. (2013). The generalized minimum spanning tree problem: A parameterized complexity analysis of bi-level optimisation. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO)*, pp. 519–526.

Deb, K., and Sinha, A. (2009). Solving bilevel multi-objective optimization problems using evolutionary algorithms. In *Proceedings of the International Conference on Evolutionary Multi-Criterion Optimization.* pp. 110–124. Lecture Notes in Computer Science, Vol. 5467.

Deb, K., and Sinha, A. (2010). An efficient and accurate solution methodology for bilevel multi-objective programming problems using a hybrid evolutionary local-search algorithm. *Evolutionary Computation*, 18(3): 403–449.

Downey, R. G., and Fellows, M. R. (1999). *Parameterized complexity*. New York: Springer.

Fischetti, M., Salazar González, J. J., and Toth, P. (1997). A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, 45(3): 378–394.

Ghosh, D. (2003). Solving medium to large sized Euclidean generalized minimum spanning tree problems. *Technical Report, Indian Institute of Management*, Research and Publication Department, Ahmedabad, India.

Hu, B., and Raidl, G. R. (2011). An evolutionary algorithm with solution archive for the generalized minimum spanning tree problem. In *Proceedings of the International Conference on Computer Aided Systems Theory.* pp. 287–294. Lecture Notes in Computer Science, Vol. 6927.

Hu, B., and Raidl, G. R. (2012). An evolutionary algorithm with solution archives and bounding extension for the generalized minimum spanning tree problem. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO),* pp. 393–400.

Koh, A. (2007). Solving transportation bi-level programs with differential evolution. In *IEEE Congress on Evolutionary Computation*, pp. 2243–2250.

Kratsch, S., Lehre, P. K., Neumann, F., and Oliveto, P. S. (2010). Fixed parameter evolutionary algorithms and maximum leaf spanning trees: A matter of mutation. In *Parallel Problem Solving from Nature*, PPSN XI, pp. 204–213.

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

Legillon, F., Liefooghe, A., and Talbi, E.-G. (2012). COBRA: A cooperative coevolutionary algorithm for bi-level optimization. In *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 1–8.

Motwani, R., and Raghavan, P. (1995). *Randomized algorithms*. Cambridge: Cambridge University Press.

Myung, Y.-S., ho Lee, C., and wan Tcha, D. (1995). On the generalized minimum spanning tree problem. *Networks*, 26(4): 231–241.

Neumann, F., and Witt, C. (2010). *Bioinspired computation in combinatorial optimization: Algorithms and their computational complexity*. New York: Springer.

Pop, P. C. (2004). New models of the generalized minimum spanning tree problem. *Journal of Mathematical Modelling and Algorithms*, 3(2): 153–166.

Sinha, A., Malo, P., and Deb, K. (2014). Test problem construction for single-objective bilevel optimization. *Evolutionary Computation*, 22(3): 439–477.

Sutton, A. M., and Neumann, F. (2012a). A parameterized runtime analysis of evolutionary algorithms for the Euclidean traveling salesperson problem. Retrieved from arXiv:1207.0578v2.

Sutton, A. M., and Neumann, F. (2012b). A parameterized runtime analysis of simple evolutionary algorithms for makespan scheduling. In *Parallel Problem Solving from Nature, PPSN XII*, pp. 52–61.