On the Impact of Local Search Operators and Variable Neighbourhood Search for the Generalized Travelling Salesperson Problem

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

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

ABSTRACT

The generalized travelling salesperson problem is an important NP-hard combinatorial optimization problem where local search approaches have been very successful. We investigate the two hierarchical approaches of Hu and Raidl [9] for solving this problem from a theoretical perspective. We examine the complementary abilities of the two approaches caused by their neighbourhood structures and the advantage of combining them into variable neighbourhood search. We first point out 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, but where a variable neighbourhood search combining them finds the optimal solution in polynomial time.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity

Keywords

Generalized Travelling Salesperson Problem; Local Search; Variable Neighbourhood Search; 2-OPT; Combinatorial Optimisation.

1. INTRODUCTION

Metaheuristics [6] such as local search, simulated annealing and various types of evolutionary algorithms have been successfully applied to a wide range of problems from combinatorial optimization. Despite their large practical success, it is very difficult to understand how and why they work on important classes of problems.

One of the best known local search approaches is the Lin-Kernighan heuristic for the well-known Travelling Salesperson problem (TSP). This heuristic is built on the classical 2-opt operator for the TSP. While extremely successful, it is still very hard to understand the success of these approaches from a theoretical perspective. Theoretical results on different types of metaheuristics for the TSP have been presented in [4, 12, 17] and the impact of different local search strategies and their combination with evolutionary algorithms on different types of problems have been analysed in [15, 16] by rigorous runtime analyses. The runtime analysis of metaheuristics has provided a lot of rigorous new insights into the working principles of these algorithms and we refer the reader to the three books [13, 1, 10] for comprehensive presentations. Despite this progress and the advancement of methods for analysing metaheuristics, understanding and analysing the run of even simple metaheuristics on classical problems such as run of local search on TSP remains challenging, and obtaining theoretical results that match practical experience is to a large extend an open problem.

With this paper, we contribute to the theoretical understanding of local search methods for the generalized travelling salesperson problem (GTSP). The problem is given by a set of cities with distances between them. Furthermore, 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. Different heuristic approaches for the GTSP have been presented in recent years [7, 14, 5, 9]. We investigate the two hierarchical approaches for solving the GTSP presented in [9]. Both approaches construct an overall solution based on an upper and lower level. The *Cluster-Based* approach uses a permutation on the clusters in the upper level and finds the best node selection based on that permutation on the lower layer. The *Node-Based* approach selects a spanning node for each cluster and then works on finding the best permutation of the chosen nodes. Hu and Raidl [9] have combined the two approaches into a variable neighbourhood search algorithm and shown that this leads to a high performing algorithm for the GTSP.

We investigate the two approaches from a theoretical perspective and point out complementary abilities. Our aim is to show situations where one of the approaches gets stuck in a local optimum and the other approach is able to perform well and achieves an optimal solution. This gives a deeper insight into the working principles of these two common approaches and highlights their complementary abilities. To gain these structural insights, our instances should be simple enough for theoretical treatment. As we are con-

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

sidering hierarchical approaches working with two solution layers, it is very difficult to argue in general about the run of metaheuristics on these problems. The only runtime analysis that we are aware of is the analysis of simple evolutionary algorithms in the context of bilevel optimization for the generalized minimum spanning tree problem [2] and the generalized travelling salesperson problem [3] in the context of parameterized complexity which shows that evolutionary algorithms using a cluster-based approach perform well for these problems if the number of clusters is small. We concentrate on instances with a small number of clusters to get further insights into hierarchical approaches for the GTSP. In particular, we present instances where the two approaches mutually outperform each other. Furthermore, we present an instance where both local search approaches are not able to achieve an optimal solution, but a combination of them into a variable-neighbourhood search reaches an optimal solution.

The outline of the paper is as follows. Section 2 introduces the problem and the algorithms that are subject to our investigations. In Section 3, we introduce a hard instance for the Cluster-Based approach which is easy to solve for Node-Based approach. Section 4 includes an instance easy for Cluster-Based approach and difficult for Node-Based approach and Section 5 introduces the third instance which is difficult for both of them but an algorithm that combines the two approaches can solve it easily. Finally, we finish with some concluding remarks in Section 6.

2. PROBLEM AND ALGORITHMS

We consider the generalized travelling salesperson problem (GTSP) which can be described as follows. The input is given by a (un)directed complete graph $G = (V, E, c)$ with $n = |V|$ nodes, a cost function $c\colon E \to \mathbb{R}^+$ on the edges, and a partitioning of V into m clusters $V_1, V_2, \ldots, V_m$. The goal is to find a cycle of minimum cost that visits exactly one node from each cluster.

A solution for this problem consists of two parts. A set of *spanning nodes* $P = \{p_1, p_2, \ldots, p_m\}$ where $p_i \in V_i$, and a Hamiltonian tour on the graph $G[P] = G(P, \{e \in E \mid e \subseteq P\})$ induced by P, i.e. the graph consisting of the spanning nodes and all edges between them. Following [9], we present a candidate solution as $S = (P, \pi)$ where $P = \{p_1, \ldots, p_m\}$ is the spanning node set and $\pi = (\pi_1, \ldots, \pi_m)$ is a permutation of the given clusters. Let p_{π_i} be the chosen node for cluster V_{π_i}, $1 \le i \le m$. Then the cost of a solution $S = (P, \pi)$ is given by

$$c(S) = c(p_{\pi_m}, p_{\pi_1}) + \sum_{i=1}^{m-1} c(p_{\pi_i}, p_{\pi_{i+1}}).$$

2.1 Cluster-Based Local Search

In the Cluster-Based approach, constructing the permutation of clusters constitutes the upper layer and the node selection is done in the lower layer.

Let $\pi = (\pi_1, \cdots, \pi_m)$ be a permutation of the m clusters. The 2-opt neighbourhood of π is given by

$$\begin{aligned} N(\pi) \;\; = \;\; & \{\pi' \mid 1 \le i < j \le m, \\ & \pi' = (\pi_1, \cdots, \pi_{i-1}, \pi_j, \pi_{j-1}, \cdots, \pi_i, \pi_{j+1}, \cdots, \pi_m)\} \end{aligned}$$

The cluster-based local search (CBLS) algorithm working with this neighbourhood structure is given in Algorithm 1.

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
8:    end if
9: end for
```

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

```
1: Choose $P = \{p_1, p_2, \ldots, p_m\}$,   $p_i \in V_i$.
2: Let $\pi$ be the permutation of clusters obtained by per-
   forming 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
8:    end if
9: end for
```

CBLS starts with an initial permutation of clusters. At each step, a new permutation π' is selected from the 2-opt neighbourhood of π, where π is the current permutation of clusters. Then the lower layer uses the shortest path algorithm given in [11] to find the best spanning node set. The runtime of the algorithm to compute an optimal set of spanning nodes for a given permutation using this algorithm is $O(n^3)$.

The new solution $S' = (P', \pi')$ replaces the old one $S = (P, \pi)$ if it is of less cost and the algorithm terminates if no better permutation can be found in the 2-opt neighbourhood of the current solution π.

2.2 Node-Based Local Search

In the Node-Based approach, selection of the spanning nodes is done in the upper layer and the lower level consists of finding a shortest tour on the spanning nodes.

Given a spanning nodes set P, the node exchange neighbourhood $N'(P)$ is defined as

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

Note that the lower level involves solving the classical TSP; therefore, poses in general an NP-hard problem on its own. For our theoretical investigations, we consider two algorithms NEN-LS and NEN-LS*. NEN-LS computes 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 programming [8] and is practical if the number of clusters is small. We use NEN-

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

```
1: Choose $P = \{p_1, p_2, \cdots, p_m\}$,   $p_i \in V_i$.
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
8:   end if
9: end for
```

Algorithm 4 Variable Neighborhood 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:     end if
10:  end for
11:  $l = l + 1$
12: end while
```

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* (see Algorithms 2 and 3) 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. We use a general variable neighbourhood scheme which explores different neighbourhood structures. The algorithm uses the two neighbourhood structures of CBLS and NEN-LS.

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

- $N_1(S) = \{S' = (P', \pi') \mid \pi' \in N(\pi), P' =$ optimal set of nodes with respect to $\pi'\}$
- $N_2(S) = \{S' = (P', \pi') \mid P' \in N'(P), \pi' =$ order of clusters obtained by 2-opt from π 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. N_1 is the first neighbourhood to be searched and N_2 is used when a local optimum has been found with respect to N_1. The resulting variable neighbourhood search (VNS) algorithm is given in Algorithm 4.

3. 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.

We consider the undirected 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 is given by the nodes of the different clusters.

The edge set E consists of 4 types of edges which we define in the following.

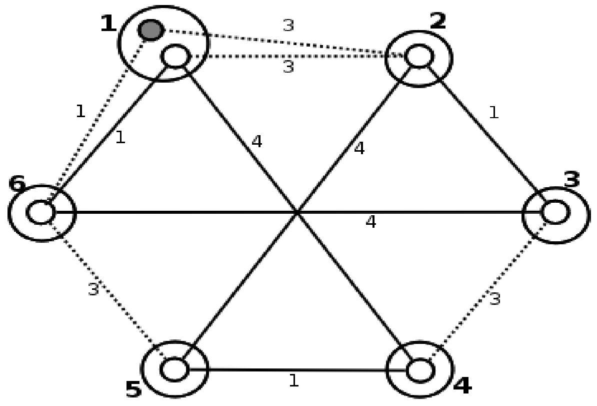

Figure 1: An instance of the problem, easy for Node-Based approach and hard for Cluster-Based approach

- Type A: Edges of this kind 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 kind.

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

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

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

Figure 1 shows the important connections of the nodes of this graph. For simplicity, we have drawn only one node for each group of similar nodes with similar edges in the picture.

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

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. Therefore, the maximum number of edges of this type that can be used in a tour is 3. The second least costly type of edge is B with weights of 3. This implies that no tour can be shorter than $3 \cdot 1 + 3 \cdot 3 = 12$. Each solution with a permutation in consecutive or reverse-consecutive order uses exactly three edges of weight 1 and three edges of weight 3 which implies a cost of 12 and is therefore optimal. □

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

Proof. Starting with all white nodes and a permutation $\pi = (1, 4, 5, 2, 3, 6)$, the 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 implies 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:
  - 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 $\pi' = (1, 5, 4, 2, 3, 6)$ which 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 3 and cluster 6. This 2-opt results in permutation $\pi' = (1, 3, 2, 5, 4, 6)$ which also adds two edges of type D to the solution, making the total cost greater than 15.
  - 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 $\pi' = (1, 4, 5, 3, 2, 6)$ which also adds two edges of type D to the solution, making the total cost greater than 15.

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 $O(nm^2)$ steps.*

Proof. Starting with a solution with only white nodes and the permutation of $\pi = (1, 4, 5, 2, 3, 6)$, the lower level is already locally optimal using the arguments in the proof of Theorem 2. This implies that the solution does not change unless a grey note 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.

Considering $\pi' = (1, 2, 5, 4, 3, 6)$, the only acceptable 2-opt move leads to the global optimum $\pi_{opt} = (1, 2, 3, 4, 5, 6)$. The runtime is bounded by $O(nm^2)$ as it takes $O(n)$ time on the upper level to selected a grey node. Furthermore, each lower level optimization is bounded by $O(m^2)$ as either permutations are locally optimal with respect to the spanning nodes or there are at most two improvements of the permutation in the case that a grey node of cluster V_1 is selected. $\square$

4. BENEFITS OF CBLS

We now consider a situation where NEN-LS* finds it hard to obtain an optimal solution and CBLS with the same 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 are more costly than edges connecting the other clusters. The edge set E of G_2 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.

$$A = \quad \{\{v_{iW}, v_{jW}\} \mid 2 \leq i, j \leq m\}$$

- Type B: Edges of this type have a weight of 2. All connections between black nodes of different clusters are of this type.

$$B = \quad \{\{v_{iB}, v_{jB}\} \mid 1 \leq i, j \leq m\}$$

- Type C: Edges of this type have a weight of m. All edges between white node of the costly cluster and white nodes of other clusters are of this type.

$$C = \quad \{\{v_{1W}, v_{iW}\} \mid 2 \leq i \leq m\}$$

- Type D: Edges of this type have a weight of m^2. All other edges in this complete graph, which consist of all edges between a white and a black node, are of this type.

$$\begin{aligned} D = & \quad E \setminus \{A \cup B \cup C\} \\ = & \quad \{\{v_{iW}, v_{jB}\} \mid 1 \leq i, j \leq m\} \end{aligned}$$

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.

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. $\square$

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

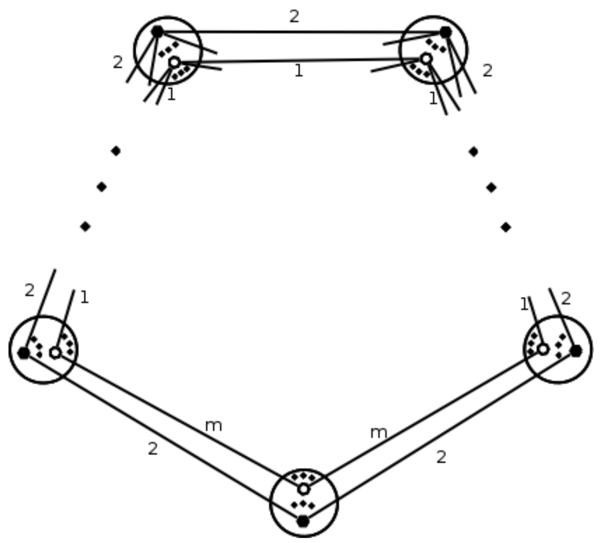

Figure 2: Graph G_2

Theorem 5. *Starting with an arbitrary permutation π, CBLS finds an optimal solution for G_2 by choosing the optimal spanning node set P for π 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. Such a set P is constructed in time $O(n^3)$ by the shortest path algorithm given in [11]. $\square$

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* even uses an exact solver for the lower layer.

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 layer selects a permutation which forms a chain of white nodes and a chain of black nodes connected to form a cycle by only two edges of type D.

Let p_1 be the selected node of the costly cluster V_1. If p_1 is initially white, the lower layer places it at one border between the black chain and the white chain to avoid using one of edges of type C. This situation is illustrated in Figure 3. If p_1 is initially black, then the initial solution would look like Figure 4.

Claim 7. *Starting with a random initial solution, 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 solution has a chain of black nodes and a chain of white nodes.

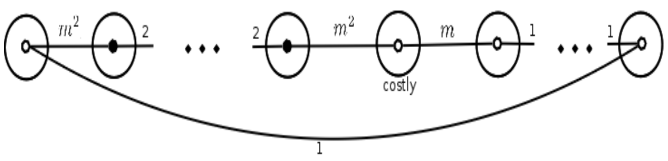

Figure 3: The initial solution for G_2 if a white node is selected for the costly cluster V_1

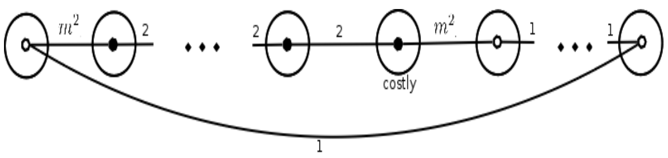

Figure 4: A possible solution for G_2 if a black node is selected for the costly cluster V_1

Changing a black node $p_i, i \neq c$ 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 $V_i, i \neq c$ a change from white to black cannot happen.

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 with removing the edge of type C while 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 4. 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 $p_i, i \neq 1$ gradually are changed to white in the local search that is performed in the NEN. For p_1:

- If p_1 is initially black, it remains black until all other p_is change to white. At this point p_1 is the only black node in the solution and is connected to two white nodes with edges of type D and cost m^2 as illustrated in Figure 5. 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 6). This change is accepted because two edges of cost m are less costly than two edges of cost m^2.
- If p_1 is initially white,

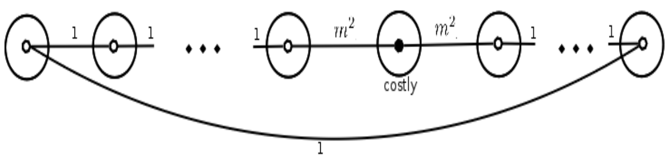

Figure 5: All other clusters change to white one by one

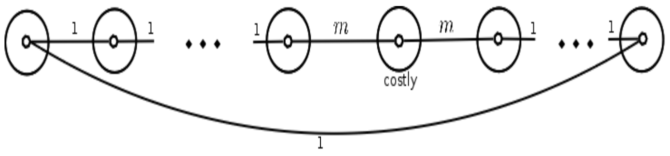

Figure 6: Local Optimum for G_2

  – If it happens to change to black, it remains black until all other p_is change to white, at which point p_1 also changes to white.
  – If all other p_is change to white before trying a black node for p_1, then it never changes to black.

This eventually results in a local optimum with all white nodes selected. The algorithm only needs to traverse the clusters on the upper layer only twice which gives $O(m)$ iterations on the upper layer for the algorithm to get stuck in a local optimum. In the first traverse, for all the clusters the white node will be selected except for the costly cluster V_1 for which the black node will be selected. In the second traverse, that only black node will also change to white. □

5. 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 results in finding the global optimum.

We consider the undirected graph G_3 shown in Figure 7 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.

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. 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.

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

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

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

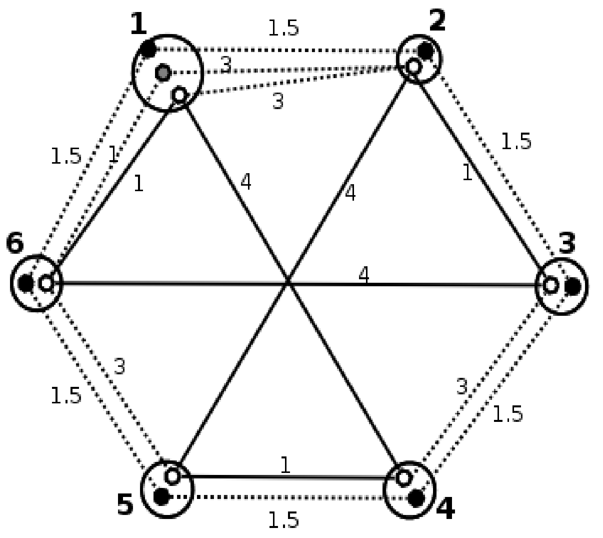

Figure 7: Graph G_3 showing one node of each type for each cluster and omitting edges of cost 100.

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

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

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

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

- Type F: Edges of this kind 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 forth 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.

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 with the optimal solution of cost 12. Therefore, visiting all black nodes with the cost of 9 is the optimal solution. □

We now show that the algorithms CBLS and NEN-LS may get stuck in a local optimum.

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 which are of type F. Therefore, any solution consisting of two kinds of nodes, black and another kind, cannot be accepted after this solution. Considering only white and grey nodes, the permutation $\pi' = (1, 2, 3, 4, 5, 6)$ is better than the current one, but as we saw in Theorem 2 of Section 3 this order can not be achieved with Algorithm 1. A solution consisting of all the black nodes is less costly only if they are visited in the optimal order of $\pi' = (1, 2, 3, 4, 5, 6)$ which is exactly the same permutation that is better for white nodes as well. As we discussed, this permutation is not achievable by searching the 2-opt neighbourhood of the current solution and the Cluster-Based approach can not find it.

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 layer finds the optimal order of clusters similar to what we described in Theorem 3 of Section 3 which form a solution of cost 12. From this point any Node-Exchange-Neighbourhood search fails to find a better solution. □

Using the combination of the two hierarchical approaches by variable-neighbourhood search allows us to escape these local optima. As a result 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. 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 layer. □

The investigations of this section have pointed out where a combination of the two hierarchical approaches into variable neighbourhood search gives a clear benefit to the optimization process as it is crucial for escaping local optima of the two single approaches.

6. CONCLUSION

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 neighbourhoods from a theoretical perspective. 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.

Acknowledgements

The research leading to these results has received funding from the Australian Research Council (ARC) under grant agreements DP130104395 and DP140103400.

7. REFERENCES

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

[2] D. Corus, P. K. Lehre, and F. Neumann. The generalized minimum spanning tree problem: a parameterized complexity analysis of bi-level optimisation. In C. Blum and E. Alba, editors, *Genetic and Evolutionary Computation Conference, GECCO '13, Amsterdam, The Netherlands, July 6-10, 2013*, pages 519–526. ACM, 2013.

[3] D. Corus, P. K. Lehre, F. Neumann, and M. Pourhassan. A parameterized complexity analysis of bi-level optimisation with evolutionary algorithms. *CoRR*, abs/1401.1905, 2014.

[4] 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.

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

[6] M. Gendreau and J.-Y. Potvin. *Handbook of Metaheuristics.* Springer Publishing Company, Incorporated, 2nd edition, 2010.

[7] G. Gutin and D. Karapetyan. A memetic algorithm for the generalized traveling salesman problem. *Natural Computing*, 9(1):47–60, 2010.

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

[14] P. C. Pop and S. Iordache. A hybrid heuristic approach for solving the generalized traveling salesman problem. In N. Krasnogor and P. L. Lanzi, editors, *GECCO*, pages 481–488. ACM, 2011.

[15] D. Sudholt. Hybridizing evolutionary algorithms with variable-depth search to overcome local optima. *Algorithmica*, 59(3):343–368, 2011.

[16] D. Sudholt. Parametrization and balancing local and global search. In F. Neri, C. Cotta, and P. Moscato, editors, *Handbook of Memetic Algorithms*, volume 379 of *Studies in Computational Intelligence*, pages 55–72. Springer, 2012.

[17] 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):595628, 2014.