The Covering Canadian Traveller Problem Revisited

Niklas Hahn
Sorbonne Université, CNRS, LIP6, F-75005 Paris, France

Michalis Xefteris
Sorbonne Université, CNRS, LIP6, F-75005 Paris, France

Abstract

In this paper, we consider the k-Covering Canadian Traveller Problem (k-CCTP), which can be seen as a variant of the Travelling Salesperson Problem. The goal of k-CCTP is finding the shortest tour for a traveller to visit a set of locations in a given graph and return to the origin. Crucially, unknown to the traveller, up to k edges of the graph are blocked and the traveller only discovers blocked edges online at one of their respective endpoints. The currently best known upper bound for k-CCTP is $O(\sqrt{k})$ which was shown in [Huang and Liao, ISAAC '12]. We improve this polynomial bound to a logarithmic one by presenting a deterministic $O(\log k)$-competitive algorithm that runs in polynomial time. Further, we demonstrate the tightness of our analysis by giving a lower bound instance for our algorithm.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Online Algorithm, Canadian Traveller Problem, Travelling Salesperson Problem, Graph Exploration

Digital Object Identifier 10.4230/LIPIcs...

Acknowledgements We would like to thank Evripidis Bampis and Bruno Escoffier for providing useful comments on the manuscript.

1 Introduction

The Canadian Traveller Problem (CTP) was introduced in 1991 by Papadimitriou and Yannakakis [26] as an extension of the Shortest Path Problem and has applications in online route planning in road networks. The goal of the problem is to find a shortest path between a source and a destination in an unreliable graph, in which some edges may become unavailable. This can only be observed in an online manner, i.e., when reaching one of the endpoints of such an edge. More specifically, consider a connected, undirected graph $G = (V, E)$ with a source node $s \in V$, a destination node $t \in V$ and a non-negative cost function $c \colon E \to \mathbb{R}^+$ representing the cost to traverse each edge. A traveller seeks to find a path with minimum cost from s to t. However, one or more edges might be blocked, and thus cannot be traversed. The traveller only learns that an edge is blocked when reaching one of its endpoints. When the number of blocked edges is bounded by k, the variant is called k-Canadian Traveller Problem or k-CTP [6].

This work studies a generalization of CTP, defined in [22], which is called the Covering Canadian Traveller Problem (CCTP). In CCTP, one attempts to develop an efficient tour for a traveller that visits all vertices in a graph and returns to the origin (source) under the same uncertainty as that of CTP. When the number of blocked edges is bounded by k, the problem is called k-CCTP, analogous to the k-CTP variant of CTP.

We make two assumptions on the underlying graph model, similar to [22]. First, we assume that the graph remains connected even if all blocked edges are removed. Second, the state of an edge, i.e., whether it is blocked or not, does not change after the traveller learns about it. The problem has practical uses in dynamic routing systems that prioritize efficient travel routes and aim to avoid traffic congestion. Since Huang and Liao introduced CCTP [22], there has been a notable amount of work on similar problems in the literature [1, 30, 36, 37]. For example, Zhang et al. [35] studied the Steiner Travelling Salesperson Problem in which the salesperson instantly learns about new blocked edges. Shiri et al. [29] focused on how to allocate and route search-and-rescue teams to areas with trapped victims, considering the uncertainty about road conditions which may delay the operations.

The motivation behind CCTP stems from other similar optimization problems, such as dynamic TSP and online TSP. Dynamic TSP has been studied for various different types of dynamic change, such as the addition or removal of locations, and changing pairwise distances between locations in the underlying space [21, 32]. On the other hand, Ausiello et al. [4] introduced the online TSP in which the input arrives over time, i.e., during the travel new requests (locations) appear that have to be visited by the algorithm. The problem has many practical applications, e.g., in logistics and robotics [2, 27]. Since its introduction, a series of papers has been published on the subject [3, 10, 19].

As is usual in the literature on online problems, we measure the performance of our algorithm by its competitive ratio [11]. This means that its performance is compared to the performance of an algorithm for the corresponding offline problem. In our setting, this would be an algorithm which knows the complete graph structure, including all blocked edges.

Our Contribution

In this paper, we focus on k-CCTP. Currently, the best known deterministic algorithm for k-CCTP is the Cyclic Routing algorithm by Huang and Liao [22] with competitive ratio $O(\sqrt{k})$. We improve this bound to $O(\log k)$ by making a connection with the Online Graph Exploration problem. In the Online Graph Exploration problem, a searcher starts from a source vertex and aims to visit all vertices of an unknown but fixed graph. Upon reaching a new vertex, the server learns all incident edges and their costs. The reduction we give allows us to get a polynomial time algorithm for k-CCTP using an algorithm for the Exploration problem. Finally, we show that our analysis of the $O(\log k)$-competitive algorithm is tight.

2 Related Work

Online Graph Exploration Problem

In the Online Graph Exploration problem, defined in [20], an agent has to explore an unknown graph by starting at a given vertex, visiting all other vertices, and returning to the starting one. The agent can only move along the edges of the graph and has to pay a cost for each traversed edge.

A simple and fast algorithm that solves the problem is the Nearest Neighbor (NN) algorithm. The algorithm selects an unexplored vertex that is cheapest to reach from the current one and visits it, repeating this process until all vertices are visited. This algorithm has been shown to have a competitive ratio of $\Theta(\log n)$ for arbitrary graphs in the Online Graph Exploration Problem [28], which is a tight bound even on planar unit-weight graphs [15, 18]. Note that although the analysis in [28] deals with the offline problem, the nearest neighbor can always be identified even in the online scenario and the same analysis applies. The second algorithm that achieves the $\Theta(\log n)$ bound, which is the best known upper bound for arbitrary graphs, is the hierarchical Depth First Search algorithm (hDFS) in [23].

On the other hand, obtaining constant-competitive tours is known only for special cases of graphs, such as graphs with k distinct weights, graphs with bounded genus, cycles, tadpole graphs and cactus graphs [12, 15, 20, 23, 24]. Conversely, the best known lower bound on the competitive ratio of an online algorithm is just $10/3$ [9], and despite efforts, it is still unclear whether there exists an $o(\log n)$ or even $O(1)$-competitive exploration algorithm for general graphs.

Canadian Traveller Problem

CTP has been proven to be PSPACE-complete [26]. For the k-CTP variant, Bar-Noy and Schieber proposed a polynomial time algorithm that minimizes the maximum travel length [6]. Westphal developed a simple deterministic online algorithm for k-CTP that is $(2k + 1)$-competitive and proved that no deterministic online algorithm can have a (strictly) better competitive ratio [33]. Furthermore, he showed a lower bound of $k + 1$ for any randomized algorithm, even if all $s - t$ paths are node disjoint. Xu et al. [34] proposed a deterministic algorithm that is also $(2k+1)$-competitive for k-CTP and proved that a natural greedy strategy based on the available blockage information is exponential in k. On graphs where all $s - t$ paths are node-disjoint, a $(k + 1)$-competitive randomized online algorithm is known [7, 31]. Demaine et al. [14] proposed a polynomial time randomized algorithm that improves the deterministic lower bound of $2k + 1$ by an $o(1)$ factor for arbitrary graphs. They also showed that the competitive ratio is even better if the randomized algorithm runs in pseudo-polynomial time. Recently, Bergé et al. [8] proved that the competitive ratio of any randomized algorithm using a specific set of strategies called memoryless cannot be better than $2k + O(1)$. Over the last few years, various other variants of CTP have been investigated [5, 17, 25].

Covering Canadian Traveller Problem

The best known algorithm for k-CCTP is the one proposed in [22] with a competitive ratio of $O(\sqrt{k})$. The algorithm, called CYCLIC ROUTING, decomposes the entire route into several rounds. In each round, the traveller attempts to visit as many vertices as possible in the graph following the visiting order (or the reverse order) of the tour derived by Christofides' algorithm [13].

3 Preliminaries

In this section, we give some basic definitions. We start by giving a formal definition of the problem we study, before defining the Online Graph Exploration Problem, the performance measure and restating Christophides' algorithm for completeness. In what follows, we will denote by $G = (V, E)$ a weighted, undirected graph. We will interchangeably use the notion of "cost" and "length" for the weight of an edge. For example, a shortest tour is a tour of minimum cost.

Definition of k-CCTP

The formal definition of CCTP is as follows. Given a complete metric graph $G = (V, E)$ with a source vertex $s \in V$, a traveller aims, beginning from s, to visit every other vertex in V at least once and return to s with as little cost as possible. However, the traveller discovers online that some edges are blocked once reaching one of their endpoints. Moreover, as mentioned earlier, two assumptions are made. First, the blocked edges cannot isolate vertices of G, i.e., G remains connected, and second, edges remain in their state (i.e., whether they are blocked or not) forever. In this paper, we consider its variant k-CCTP where the number of blocked edges is bounded by k.

Definition of the Online Graph Exploration Problem

The problem can be formalized as follows. Let $G = (V, E)$ be a weighted, undirected graph with $|V| = n$ vertices. The agent starts at a vertex $s \in V$ and has to visit every vertex in the graph and return to s. Note that the agent can visit a vertex more than once. At each step, the agent is located at a vertex u and can choose to move to any of the neighboring vertices of u. The agent incurs a cost equal to the cost of the edge traversed. Upon arriving at a vertex v, the agent learns all the edges incident to v and their costs.

Competitive Ratio

A deterministic online algorithm ALG for k-CCTP is c-competitive if the total cost $|ALG(\sigma)|$ accrued by ALG for input σ is at most $c \cdot |OPT(\sigma)|$. Here, $|OPT(\sigma)|$ is the total cost of an optimal tour for σ which is computed by an offline algorithm that already knows all blocked edges.

Christophides' algorithm

We also remind the reader how Christophides' algorithm works. Christophides' algorithm on a complete metric graph G can be described as follows [16]:

1. Create a minimum spanning tree T of G.
2. Find a minimum-weight perfect matching M in the subgraph of G that is induced by the vertices with odd degrees in T.

3. Combine the edges of M and T to form a connected multigraph H.
4. Form a Eulerian cycle in H.
5. Make the circuit found in the previous step into a Hamiltonian cycle by skipping repeated vertices.

4 Solving k-CCTP via Graph Exploration

In this section, we present the results of our work. First, we show a connection between CCTP and the Online Graph Exploration problem (Theorem 1). This is the crucial step to improve the upper bound of $O(\sqrt{k})$.

The idea behind our reduction is that CCTP can be solved by an algorithm that solves the Online Graph Exploration Problem. This is possible since at every step, the traveller locally learns the real edges in both problems. The challenge here is that the algorithms for the Online Graph Exploration for arbitrary graphs have competitive ratios depending on the number of vertices n.

So, how can we reduce the size of the graph in which we run an algorithm for Graph Exploration to something of size $O(k)$? First, we try to follow an approximately optimal TSP tour, skipping vertices when edges are discovered to be blocked. Similar to the idea in Cyclic Routing of [22], we use a function ShortCut to achieve this. After that, we return to the starting vertex. This way, we visit at least $n-k$ vertices of G. Since they do not have to be visited again, we can then use the information gathered to reduce the number of vertices in the graph on which we will run the algorithm for Graph Exploration to $O(k)$. Formally, we have the following theorem.

▶ **Theorem 1.** *If there exists an $f(k)$-competitive algorithm for the Online Graph Exploration problem on graphs with at most $k+1$ nodes and an α-approximation algorithm for metric TSP, then there exists an $(f(k)+2\alpha)$-competitive algorithm for k-CCTP.*

Proof. Suppose that we have AlgoExploration, an $f(k)$-competitive algorithm for the Online Graph Exploration problem on arbitrary graphs with at most $k+1$ nodes, and AlgoTSP, an α-approximation algorithm for metric TSP. Then, we will prove that the algorithm CompressAndExplore that uses these algorithms as subroutines has a competitive ratio of at most $f(k)+2\alpha$ for k-CCTP (for pseudocode, see Algorithm 1).

First, the algorithm runs AlgoTSP on the input graph G to compute a TSP tour P. For simplicity, we relabel the vertices with respect to the tour, i.e., we assume that the tour P has the order $s=v_1 \to v_2 \to \cdots \to v_n \to v_1$. If an edge $\{v_i, v_j\}$ is blocked, the traveller tries to go to the next vertex in the order determined by P, i.e., v_{j+1}, or v_1 for $j=n$ (for pseudocode of this subroutine, see Function ShortCut on page 8). This procedure is possible since the original graph is complete. By the triangle inequality, the cost of the tour is upper bounded by the cost of tour P. If the traveller reaches vertex s, then ShortCut terminates. If s is not reachable directly because of a blocked edge, the traveller returns to s by retracing their steps. Since cost(P) is an α-approximation for metric TSP, we have that cost(ShortCut) $\leq 2 \cdot \text{cost}(P) \leq 2\alpha \cdot |OPT|$, where OPT is an optimal offline TSP tour on graph G.

The traveller learns about all blocked edges which are adjacent to the vertices that are visited during ShortCut. In the procedure, all edges that are discovered to be blocked are collected in the set E_b. Thus, the traveller knows the whole graph (with all blocks) except for the induced (complete) subgraph formed by the unvisited vertices U. Let κ be the number of vertices which remain unvisited by ALG after ShortCut, i.e., the size of the set U. Then, the traveller has discovered at least κ blocked edges, i.e., $|E_b| \geq \kappa$.

Next, the traveller, being at s, has to visit the vertices in U. Since the true edges of the graph except for those of the induced subgraph formed by vertices in U are known, it suffices to consider only the vertices in the set $U_s = U \cup \{s\}$. While the vertices in $V \setminus U_s$ themselves are not required, a shortest path between two vertices $x, y \in U_s$ might include vertices from the set $V \setminus U_s$ as intermediate nodes. This can occur when currently unknown edges between unvisited nodes are blocked. More specifically, the algorithm runs the function Compress (for pseudocode, see Function Compress on page 9). For every pair of vertices $x, y \in U_s$, the function creates a new edge $P_{x,y}$ representing a shortest path between x and y such that the path consists only of edges that are known not to be blocked, i.e., edges in which at least one node has already been visited before – if such a shortest path exists. Note that this phase of the algorithm does not incur any cost in terms of competitive ratio. Thus, the procedure creates a multigraph G' which consists of vertex set U_s, the initial edges that connect these vertices and the "shortest-path" edges as described above. To better explain the steps of the algorithm we present an example of an execution of algorithm CompressAndExplore below (see Example 1).

Finally, the algorithm runs AlgoExploration on G' and visits the remaining vertices.[1] Every time the traveller visits a vertex, they learn all incident edges. This includes the newly added "shortest-path" edges, of which we know that they are feasible. If the traveller uses such a "shortest-path" edge, then in the final computed tour in the original graph we expand it, meaning that we use the real path that corresponds to this edge. The cost of an optimal TSP tour $OPT_{G'}$ on multigraph G' is at most the cost of an optimal TSP tour $OPT_G(U_s)$ that only has to visit the vertex set U_s, i.e., the vertices that are also in G', but inside the input graph G. To see that this holds, consider an optimal tour $OPT_G(U_s)$. Assume that it visits the vertices in U_s in the order $s = x_1 \to x_2 \to \cdots \to x_{|U_s|} \to x_1$. Between any vertices $x_i, x_{i+1} \in U_s$ for $i \in \{1, \ldots, |U_s| - 1\}$ (or $x_{|U_s|}, x_1$) that are visited one after the other, $OPT_G(U_s)$ uses a shortest path. Each of these shortest paths starts in U_s. If it then uses an edge to another vertex in U_s, this edge will also be in G' as each direct edge between two vertices of U_s will either be blocked in both G and G' or not be blocked in both. Hence, we can assume that an edge $\{u, v\}$ from $u \in U_s$ to a vertex $v \notin U_s$ is taken. This is an already discovered edge, as v has already been visited during ShortCut. Eventually, the path will re-enter into the set U_s by using another already discovered edge $\{v', u'\}$ for some $v' \notin U_s$ and $u' \in U_s$. In between leaving and re-entering, all edges that were taken are also already discovered and this partial path has exactly the same length as the shortest-path edge $P_{u,u'}$ between $u, u' \in U_s$.[2] Continuing this argument, eventually the target vertex in U_s is reached. All intermediate partial paths are inside G', either since they are regular edges that also exist in G, or since they have been added as shortest-path edges during ShortCut.

The multigraph G' has $\kappa + 1$ vertices and at least κ blocks have already been discovered. The number of blocked edges is at most k, and thus there are at most $k+1$ vertices in G'. So, from the hypothesis the cost incurred by AlgoExploration on G' is at most $f(k) \cdot |OPT_{G'}|$. Since an optimal solution for visiting a subset of vertices $OPT_G(U_s)$ has cost at most $|OPT|$,

[1] AlgoExploration solves the Exploration problem on arbitrary graphs, but G' is a multigraph with at most two edges per pair of vertices. However, this does not cause a problem, since the algorithm can always select a shortest edge out of the two and the optimal solution can be computed while keeping only one edge per pair.

[2] There might be several shortest paths with the same length, which is why the shortest path chosen for $P_{u,u'}$ and the described shortest path might differ. Still, their lengths are equal by definition.

Algorithm 1 CompressAndExplore(AlgoTSP, AlgoExploration)

```
Input    : A complete metric graph $G = (V, E)$ with $n$ vertices; a starting vertex
           $s \in V$;
Output   : A tour that visits every vertex in $V$;
Parameter: AlgoTSP($G_1$): An algorithm that returns a TSP tour on a metric
           graph $G_1$; The tour has the form $s = v_1 \to v_2 \to \cdots \to v_n \to v_1$;
           AlgoExploration($G_2$): An algorithm that solves the Online Graph
           Exploration problem on an arbitrary graph $G_2$ and returns a tour;
1 $P \leftarrow$ AlgoTSP($G$);
2 $G^*, U, P_1 \leftarrow$ ShortCut($G, P$);
3 $G' \leftarrow$ Compress($G^*, U, G$);
4 $P_2 \leftarrow$ AlgoExploration($G'$);
5 $P' \leftarrow (P_1 \to P_2)$;
  /* Concatenate $P_1$ and $P_2$, i.e., visit the vertices according to $P_1$,
     then according to $P_2$.                                                */
6 return $P'$;
```

we get the following

$$\mathrm{cost}(\textsc{AlgoExploration}) \leq f(k) \cdot |OPT_{G'}| \leq f(k) \cdot |OPT_G(U_s)| \leq f(k) \cdot |OPT| .$$

Overall, the algorithm has a total cost for the traveller of

$$\begin{aligned}\mathrm{cost}(\textsc{CompressAndExplore}) &= \mathrm{cost}(\textsc{ShortCut}) + \mathrm{cost}(\textsc{AlgoExploration}) \\ &\leq (f(k) + 2\alpha) \cdot |OPT| .\end{aligned}$$

Consequently, CompressAndExplore is an $(f(k) + 2\alpha)$-competitive algorithm for k-CCTP. Note that the knowledge of k does not affect the performance of the algorithm. ◀

▶ Remark. In the proof, we allow $k \geq n - 1$ as long as the resulting graph remains connected. The analysis of the competitive ratio of $O(\sqrt{k})$ in [22] requires $k < n - 1$.

▶ **Example 1.** *Fig. 1 shows an example of* CompressAndExplore*. The traveller begins at vertex $s = v_1$ and moves in a counterclockwise direction. The given TSP tour by* AlgoTSP *here is $v_1 \to v_2 \to \cdots \to v_{16} \to v_1$. The solid lines represent the tour that the traveller follows during* ShortCut *due to the discovered blocked edges (red dashed lines). The traveller follows the shortcut path $v_1 \to v_2 \to v_4 \to v_5 \to v_9 \to v_{10} \to v_{11} \to v_{14} \to v_{16}$ and after visiting vertex v_{16}, they return back to s following the same path backwards. Next, the algorithm runs* Compress *and gets G'. Multigraph G' contains s, the remaining unvisited vertices and at most two edges between each pair of these vertices. Between v_i and v_j there is the edge $\{v_i, v_j\}$ (which may be blocked) and possibly the "shortest-path" edge $P_{i,j}$. The cost of $P_{i,j}$ is the cost of the shortest path from v_i to v_j in which each edge has at least one endpoint outside of G'. In the example, a possible case for $i = 1$ and $j = 3$ is shown on the right with $P_{1,3}$ being the path $v_1 \to v_4 \to v_3$.*

Finally, the algorithm runs AlgoExploration *on G'. The traveller visits all remaining vertices, returns to s and the algorithm terminates.*

Now, we can use CompressAndExplore with Christophides' algorithm for metric TSP and the Nearest Neighbor for the Online Graph Exploration problem. Since this algorithm uses Christophides' algorithm and then Nearest Neighbor, we refer to it by `CNN`.

```
Function SHORTCUT($G$, $P$):
    /* $G$ is the input graph and $P$ a TSP tour.                          */
    /* $P$ has the form $s = v_1 \to v_2 \to \cdots \to v_n \to v_1$.      */
    $i \leftarrow 1$; $j \leftarrow 2$;
    $U_s \leftarrow \{s\}$; $E_b \leftarrow \emptyset$; $P' \leftarrow \{s\}$;
    /* Path $P'$ which the traveller follows is built.                     */
    while $j \leq n$ do
        Add all newly discovered blocked edges $\{v_i, x\}$, with $x \in V \setminus \{v_i\}$, to $E_b$;
        if $\{v_i, v_j\}$ is not blocked then
            $P' \leftarrow (P' \to v_j)$;                      // Append $v_j$ to $P'$
            $i \leftarrow j$;
        else
            $U_s \leftarrow U_s \cup \{v_j\}$;
        end
        $j \leftarrow j + 1$;
    end
    if $\{v_i, v_1\}$ is blocked then
        Return to $s$ following $P'$ backwards;
        $P' \leftarrow$ Concatenate the path $P'$ and the reverse of $P'$ to return to $s$;
    else
        $P' \leftarrow (P' \to v_1)$;
    end
    $G^* \leftarrow (V, E \setminus E_b)$;
    return $G^*, U_s, P'$;
end Function
```

▶ **Corollary 2.** *CNN has a competitive ratio of $O(\log k)$ for k-CCTP.*

Proof. Christophides' algorithm gives a 3/2-approximation for metric TSP. On the other hand, NN yields a competitive ratio of $O(\log n)$ for the Graph Exploration problem in an arbitrary graph, where n is the number of vertices in the graph. Thus, from Theorem 1 we get that CNN has a competitive ratio of $3 + O(\log(k + 1)) = O(\log k)$. ◀

To demonstrate that the above analysis is tight, the following theorem presents a family of instances that achieves a competitive ratio of $\Omega(\log k)$ and therefore proves the analysis to be tight.

▶ **Theorem 3.** *There exists a family of instances for which CNN has a competitive ratio of the $\Omega(\log k)$.*

Proof. We will use the graph presented in [18] to lower bound the competitive ratio of the Nearest Neighbor algorithm. For an integer $p \geq 1$, the graph $G_p = (V_p, E_p)$ consists of a chain of $2^p - 1$ triangles. G_p has 2^p vertices in its lower level, and $2^p - 1$ vertices in its upper level. The left-most vertex in the lower level is denoted by l_p, the right-most by r_p and the central vertex in the upper level is denoted by m_p. All edges in G_p have an equal cost of 1.

For our instance, we slightly modify the graph by adding another vertex u to the left of l_p with an edge $\{u, l_p\}$ of cost 1. We also set $s = l_p$ as the starting vertex. Since the input for k-CCTP is a complete graph, we also need to add some more edges. All edges from u to

```
1  Function Compress($G^*$, $U_s$, $G$):
       /* $G^*$ is the graph without the discovered blocked edges and $U_s$ the
          set of the remaining unvisited vertices in the graph together
          with the starting vertex $s$.                                   */
2      $E' \leftarrow \{\{x,y\} \in E \mid x,y \in U_s\}$;
       /* $E'$ is the subset of edges with unknown state, i.e., of $\{x,y\}$
          with $x,y \in U_s$.                                             */
3      $G' \leftarrow (U_s, E')$;
4      $H \leftarrow (V, E \setminus E')$;
       /* $H$ includes all edges with a known state, since in every edge at
          least one vertex has already been visited.                      */
5      Let $U_s = \{v'_1, v'_2, \dots, v'_{|U_s|}\}$;
6      for $i \leftarrow 1$ to $|U_s|$ do
7          for $j \leftarrow i+1$ to $|U_s|$ do
8              Find a shortest path $P_{i,j}$ from $v'_i$ to $v'_j$ in $H$;
9              $c_{i,j} \leftarrow$ total cost of $P_{i,j}$;
10             Add an edge $\{v'_i, v'_j\}$ with cost $c_{i,j}$ to $G'$;
11         end
12     end
13     return $G'$;
14 end Function
```

the other vertices have a cost of 1, and all other new edges have a cost of 2. All these edges will be blocked edges. We call this new graph G_p^+.

The resulting graph has $k = \Theta(n^2)$ blocked edges and clearly satisfies the triangle inequality. We illustrate the non-blocked part of G_p^+ for $p = 3$ in Fig. 2.

In the first step of Christophides' algorithm, a minimal spanning tree is constructed. One possible MST T is a path from u to r_p. The nodes with uneven degree in T are the nodes u and r_p, so for the matching, the edge between u and r_p is added. This results in a simple cycle of all nodes. The MST and the matching edge are illustrated in Fig. 3.

The TSP-tour can then be chosen to be $s = l_3 \to u \to r_3 \to \cdots \to l_3$. This means that in ShortCut, only node u would be visited besides l_p as the direct edges from u to any other node (besides l_p) are blocked. At the end of ShortCut, the traveller returns to l_p.

After ShortCut, the remaining graph would thus be the original graph G_p from [18]. We use the following lemma to prove that there exists a TSP-tour in G_p which starts (and ends) in l_p which is found by NN that has a length of $(p+4) \cdot 2^{p-1} - 2$. We will prove the lemma below.

▶ **Lemma 4** (Based on [18, Lemma 1]). *There exists a NN-based TSP tour on G_p which starts in l_p and visits m_p as final vertex before returning back to l_p. The tour has length $(p+4) \cdot 2^{p-1} - 2$.*

Using this result, the total cost of the described TSP-tour is $(p+4)2^{p-1}$, whereas an optimal TSP-tour has cost $2 + 3(2^p - 1)$, namely visiting u and optimally visiting G_p by going in a zig-zag motion from left to right (as shown in the MST in Fig. 3) and returning using the lower edges of the triangle, thereby using each edge of the triangles exactly once.

Figure 1 An example of algorithm CompressAndExplore.

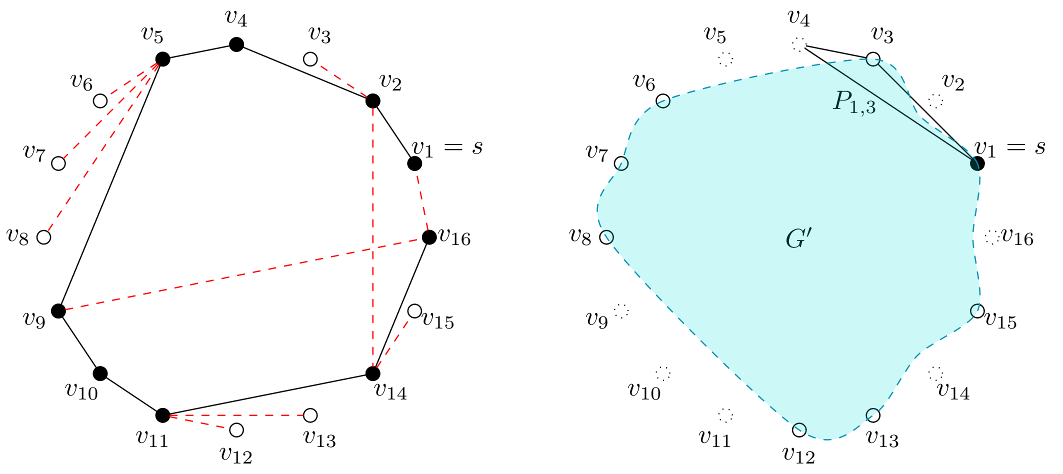
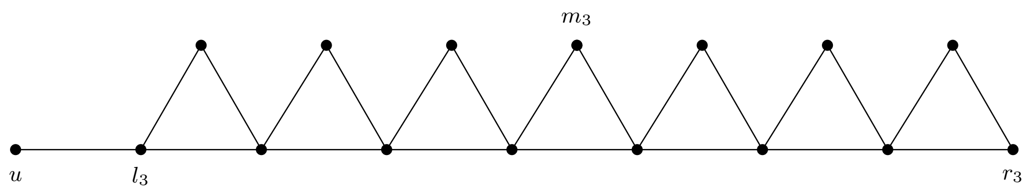


Figure 2 Graph G_p^+ (for $p = 3$). G_p^+ is used to show tightness of the $O(\log k)$-competitive ratio.



This gives us a ratio of

$$\frac{(p+4)\cdot 2^{p-1}}{2+3\cdot(2^p-1)} = \frac{(p+4)\cdot 2^{p-1}}{6\cdot 2^{p-1}-1} \geq \frac{p+4}{6} = \Omega(p) = \Omega(\log n) \ .$$

◀

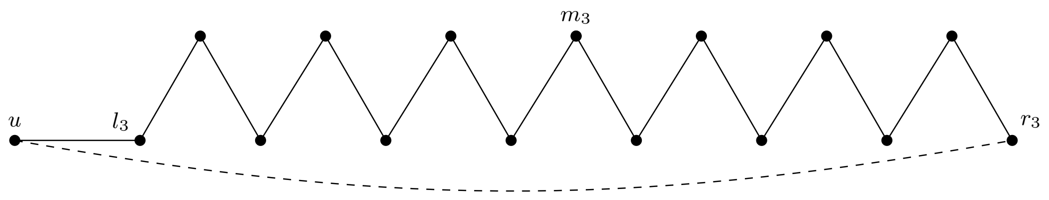
Proof of Lemma 4. We split the tour into two parts. In the first part, all vertices are visited, and in the second part, the traveller returns to l_p.

The second part has length $1 + 2^{p-1} - 1 = 2^{p-1}$. This is true because the whole graph has been discovered and the traveller can take the shortest path from m_p to l_p, which is going down to the left point of the middle triangle and then traversing the $2^{p-1} - 1$ many triangles on the left side to reach l_p.

Hence, to show the Lemma, it remains to show that there exists a NN-route to visit all vertices in G_p which has length $(p+3)\cdot 2^{p-1} - 2 = (p+4)\cdot 2^{p-1} - 2 - 2^{p-1}$, starting at l_p and ending at m_p. We prove this by induction. For $p = 1$, G_p consists of a single triangle, and the route $l_1 \to r_1 \to m_1$ has length $2 = (1+3)\cdot 2^{1-1} - 2$.

For the inductive step, we observe that G_p can be constructed from two copies of G_{p-1} and an additional vertex m_p (and three additional edges). Let $G_{p-1}^{(l)}$ be the left copy and $G_{p-1}^{(r)}$ be the right copy. Then, the new edges are $\{r_{p-1}^{(l)}, l_{p-1}^{(r)}\}, \{r_{p-1}^{(l)}, m_p\}$ and $\{l_{p-1}^{(r)}, m_p\}$. This is illustrated in Fig. 4. By the induction hypothesis, there exists an NN-route in $G_{p-1}^{(l)}$ starting in $l_p = l_{p-1}^{(l)}$ and ending in $m_{p-1}^{(l)}$ with length $(p-1+3)\cdot 2^{p-1-1} - 2$. The two nearest unvisited neighbors to $m_{p-1}^{(l)}$ are m_p and $l_{p-1}^{(r)}$ with equal distance $2^{p-2}+1$. By going to $l_{p-1}^{(r)}$, the sub-route from $l_{p-1}^{(r)}$ to $m_{p-1}^{(r)}$ of length $(p+2)\cdot 2^{p-2} - 2$ can then be found by NN. Note

Figure 3 The MST and the matching edge (dashed line) in G_3^+.



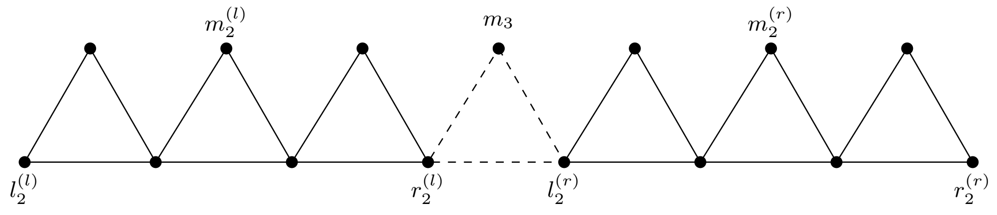
that throughout this route, m_p will never be closer to the current vertex than any other unvisited vertex in the current sub-route and thus will not be visited before $m_{p-1}^{(r)}$. Finally, m_p needs to be visited, which requires an additional cost of $2^{p-2}+1$. Overall, there exists an NN-route from l_p to m_p with length

$$2 \cdot ((p+2) \cdot 2^{p-2} - 2) + 2 \cdot (2^{p-2}+1) = (p+2) \cdot 2^{p-1} - 4 + 2^{p-1} + 2 = (p+3) \cdot 2^{p-1} - 2 \ .$$

This concludes the proof.

◀

Figure 4 Graph G_p (for $p=3$) constructed from two copies of G_{p-1}, with the joining edges denoted by the dashed lines.



Finally, we remark that `CNN` takes polynomial time. The procedures ShortCut and Compress run in polynomial time as the required shortest paths can be computed in polynomial time. Since Christophides' algorithm and Nearest Neighbor also have polynomial time complexity, so does `CNN`.

5 Concluding Remarks

In this work, we considered the Covering Canadian Traveller Problem with up to k blocked edges. We improved the upper bound to $O(\log k)$ by drawing an interesting connection to the Online Graph Exploration problem. Further, we showed the tightness of our analysis.

Our reduction implies immediate consequences of future work on the respective other problem. For one, it allows an improvement of the lower bound on the Graph Exploration problem using a general lower bound on k-CCTP. Currently, the best known bound for the Graph Exploration problem is 10/3. Tightening this gap would be a very interesting result. Second, an improved algorithm for the Graph Exploration problem immediately gives rise to a better algorithm and upper bound on k-CCTP.

Nevertheless, already an improved algorithm for k-CCTP or a lower bound on the Graph Exploration problem would be of independent interest without exploiting our reduction and thus provides another challenging direction of future research.

References

1 Vahid Akbari and Davood Shiri. Weighted online minimum latency problem with edge uncertainty. *Europ. J. Oper. Res.*, 295(1):51–65, 2021.

2 Norbert Ascheuer, Martin Grötschel, Sven O. Krumke, and Jörg Rambau. Combinatorial online optimization. In *Operations Research Proceedings 1998*, pages 21–37, 1999.

3 Giorgio Ausiello, Marc Demange, Luigi Laura, and Vangelis Paschos. Algorithms for the on-line quota traveling salesman problem. *Inf. Process. Lett.*, 92(2):89–94, 2004.

4 Giorgio Ausiello, Esteban Feuerstein, Stefano Leonardi, Leen Stougie, and Maurizio Talamo. Algorithms for the on-line travelling salesman. *Synthetic Metals*, 1999.

5 Evripidis Bampis, Bruno Escoffier, and Michalis Xefteris. Canadian traveller problem with predictions. In Parinya Chalermsook and Bundit Laekhanukit, editors, *Approximation and Online Algorithms*, pages 116–133, Cham, 2022. Springer International Publishing.

6 Amotz Bar-Noy and Baruch Schieber. The canadian traveller problem. In *Proc. 2nd Symp. Disc. Algorithms (SODA)*, pages 261–270, 1991.

7 Marco Bender and Stephan Westphal. An optimal randomized online algorithm for the k-canadian traveller problem on node-disjoint paths. *J. Comb. Opt.*, 30, 07 2013.

8 Pierre Bergé, Julien Hemery, Arpad Rimmel, and Joanna Tomasik. On the competitiveness of memoryless strategies for the k-Canadian Traveller Problem. In *Int. Conf. Comb. Opt. Appl. (COCOA)*, December 2018.

9 Alexander Birx, Yann Disser, Alexander V. Hopp, and Christina Karousatou. An improved lower bound for competitive graph exploration. *Theor. Comp. Sci.*, 868:65–86, 2021.

10 Antje Bjelde, Jan Hackfeld, Yann Disser, Christoph Hansknecht, Maarten Lipmann, Julie Meißner, Miriam Schlöter, Kevin Schewior, and Leen Stougie. Tight bounds for online TSP on the line. *ACM Trans. Algorithms*, 17(1):3:1–3:58, 2021.

11 Allan Borodin and Ran El-Yaniv. Online computation and competitive analysis. In *Cambridge University Press*, 1998.

12 Sebastian Brandt, Klaus-Tycho Foerster, Jonathan Maurer, and Roger Wattenhofer. Online graph exploration on a restricted graph class: Optimal solutions for tadpole graphs. *Theor. Comp. Sci.*, 839:176–185, 2020.

13 Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. *Operations Research Forum*, 3, 1976.

14 Erik D. Demaine, Yamming Huang, Chung-Shou Liao, and Kunihiko Sadakane. Approximating the canadian traveller problem with online randomization. *Algorithmica*, 83(5):1524–1543, 2021.

15 Robin Fritsch. Online graph exploration on trees, unicyclic graphs and cactus graphs. *Inf. Process. Lett.*, 168:106096, 2020.

16 Michael T. Goodrich and Roberto Tamassia. *Algorithm design and applications*. Wiley, 2015.

17 Yamming Huang and Chung-Shou Liao. The canadian traveller problem revisited. In *Proc. 23rd Int. Symp. Algorithms and Comp. (ISAAC)*, pages 352–361, 2012.

18 Cor A.J. Hurkens and Gerhard Woeginger. On the nearest neighbor rule for the traveling salesman problem. *Oper. Res. Lett.*, 32(1):1–4, 2004.

19 Patrick Jaillet and Xin Lu. Online Traveling Salesman Problems with Flexibility. In *Models and Algorithms for Optimization in Logistics*, volume 9261 of *Dagstuhl Seminar Proceedings (DagSemProc)*, pages 1–4, 2009.

20 Bala Kalyanasundaram and Kirk R. Pruhs. Constructing competitive tours from local information. *Theor. Comp. Sci.*, 130(1):125–138, 1994.

21 Allan Larsen, Oli Madsen, and Marius Solomon. The a priori dynamic traveling salesman problem with time windows. *Transp. Sci.*, 38:459–472, 11 2004.

22 Chung-Shou Liao and Yamming Huang. The covering canadian traveller problem. *Theor. Comp. Sci.*, 530:80–88, 2014.

23 Nicole Megow, Kurt Mehlhorn, and Pascal Schweitzer. Online graph exploration: New results on old and new algorithms. *Theor. Comp. Sci.*, 463:62–72, 2012.

24 Shuichi Miyazaki, Naoyuki Morimoto, and Yasuo Okabe. The online graph exploration problem on restricted graphs. *IEICE Transactions*, 92-D:1620–1627, 09 2009.

25 Evdokia Nikolova and David R. Karger. Route planning under uncertainty: The canadian traveller problem. In *Proc. 23rd Conf. Artif. Intell. (AAAI)*, pages 969–974, 2008.

26 Christos H. Papadimitriou and Mihalis Yannakakis. Shortest paths without a map. *Theor. Comput. Sci.*, 84(1):127–150, 1991.

27 Harilaos N. Psaraftis, Marius M. Solomon, Thomas L. Magnanti, and Tai-Up Kim. Routing and scheduling on a shoreline with release times. *Manag. Sci.*, 36(2):212–223, 1990.

28 Daniel J. Rosenkrantz, Richard E. Stearns, and Philip M. Lewis. *An analysis of several heuristics for the traveling salesman problem*, pages 45–69. Springer Netherlands, 2009.

29 Davood Shiri, Vahid Akbari, and F. Sibel Salman. Online routing and scheduling of search-and-rescue teams. *OR Spectrum*, 42:1–30, 09 2020.

30 Davood Shiri and F. Sibel Salman. On the online multi-agent O–D k-Canadian Traveler Problem. *J. Comb. Opt.*, 34, 08 2017.

31 Davood Shiri and F. Sibel Salman. On the randomized online strategies for the k-canadian traveler problem. *J. Comb. Opt.*, 38:254–267, 2019.

32 Alejandro Toriello, William Haskell, Michael Poremba, and Daniel Epstein. A dynamic traveling salesman problem with stochastic arc costs. *Oper. Res.*, 62, 10 2014.

33 Stephan Westphal. A note on the k-canadian traveller problem. *Inf. Process. Lett.*, 106(3):87–89, 2008.

34 Yinfeng Xu, Maolin Hu, Bing Su, Binhai Zhu, and Zhijun Zhu. The canadian traveller problem and its competitive analysis. *J. Comb. Optim.*, 18(2):195–205, 2009.

35 Huili Zhang, Weitian Tong, Yinfeng Xu, and Guohui Lin. The steiner traveling salesman problem with online edge blockages. *Europ. J. Oper. Res.*, 243(1):30–40, 2015.

36 Huili Zhang, Weitian Tong, Yinfeng Xu, and Guohui Lin. The steiner traveling salesman problem with online advanced edge blockages. *Computers & Operations Research*, 70:26–38, 2016.

37 Yubai Zhang, Zhao Zhang, Zhaohui Liu, and Qirong Chen. An asymptotically tight online algorithm for m-steiner traveling salesman problem. *Inf. Process. Lett.*, 174:106177, 2022.