Exact and Approximation Algorithms for Routing a Convoy Through a Graph

Martijn van Ee
Netherlands Defence Academy, Den Helder, The Netherlands

Tim Oosterwijk
Vrije Universiteit Amsterdam, The Netherlands

René Sitters
Vrije Universiteit Amsterdam, The Netherlands

Andreas Wiese
Technische Universität München, Germany

Abstract

We study routing problems of a convoy in a graph, generalizing the shortest path problem (SPP), the travelling salesperson problem (TSP), and the Chinese postman problem (CPP) which are all well-studied in the classical (non-convoy) setting. We assume that each edge in the graph has a length and a speed at which it can be traversed and that our convoy has a given length. While the convoy moves through the graph, parts of it can be located on different edges. For safety requirements, at all time the whole convoy needs to travel at the same speed which is dictated by the slowest edge on which currently a part of the convoy is located. For Convoy-SPP, we give a strongly polynomial time exact algorithm. For Convoy-TSP, we provide an $O(\log n)$-approximation algorithm and an $O(1)$-approximation algorithm for trees. Both results carry over to Convoy-CPP which – maybe surprisingly – we prove to be NP-hard in the convoy setting. This contrasts the non-convoy setting in which the problem is polynomial time solvable.

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

Keywords and phrases approximation algorithms, convoy routing, shortest path problem, traveling salesperson problem

Digital Object Identifier 10.4230/LIPIcs.MFCS.2023.86

1 Introduction

A fundamental setting within combinatorial optimization is to compute a route within a given graph, e.g., for a car in a street network. Well-studied problems in this area include the shortest path problem (SPP), the traveling salesperson problem (TSP), and the Chinese postman problem (CPP). For a single car, they model very well the problems of finding a route from one point to another, finding a tour that visits all given cities, or finding a tour that traverses all given edges, respectively.

However, this changes when instead we have a convoy that consists of several vehicles. Different streets in a network may allow different speeds. Because of safety considerations, the vehicles in the convoy need to adhere a constant inter-vehicle distance. Therefore, all the vehicles in the convoy need to travel at the same speed at any point in time. Hence, the *whole* convoy must move at the speed of the vehicle that is currently on the *slowest* edge (among all vehicles), see Figure 1. Thus, the classical algorithms for shortest path, TSP, and CPP cannot be used to find the fastest route for a convoy for these respective settings and they do not even yield approximation algorithms with any non-trivial approximation guarantee. Even more, shortest path and CPP are solvable in polynomial time in the usual setting, but it is not clear whether this is also the case for a convoy. Therefore, in this paper we investigate shortest path, TSP, and CPP in the convoy setting, present the first algorithms for these problems, and settle their complexity.

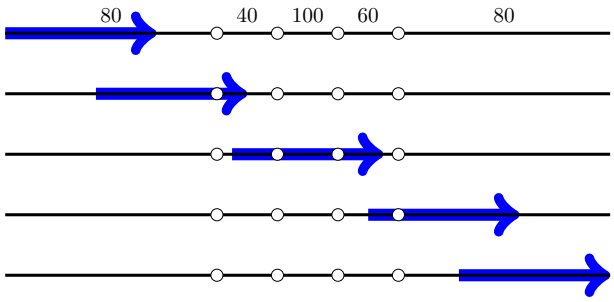

Figure 1 Five snapshots of a convoy moving along a path. The numbers along the edges denote the corresponding speeds. In the first snapshot, the convoy is entirely on an edge with speed 80, so it will move at speed 80. In the second snapshot, the front of the convoy is on an edge with speed 40, so the entire convoy will move at speed 40. In the third snapshot, the convoy still moves at speed 40. Note that in this example the convoy will never move at speed 100. In the fourth snapshot, the back of the convoy is on an edge with speed 60, so the convoy will move at speed 60. Finally, in the last snapshot, the convoy moves at speed 80.

Formally, we assume that we are given a directed graph $G = (V, E)$, where each edge $e \in E$ has a length $\ell(e) \geq 0$ and a speed $q(e) > 0$, and a length of the convoy $L \geq 0$. In Convoy-SPP we are given additionally two special vertices $s, t \in V$ and the goal is to find a path from s to t that minimizes the time between the moment that the head of the convoy leaves s until the moment that the tail arrives in t. Figuratively, we assume that at the beginning all vehicles of the convoy are located in s. Therefore, when the head of the convoy leaves s, we assume that the parts of the convoy not having left s do not restrict the speed of the convoy. We make a symmetric assumption when the convoy enters t. In Convoy-TSP we are given a special vertex $r \in V$ in which the convoy needs to start and end and we seek a tour for the convoy that visits every vertex $v \in V$ at least once, minimizing the time between the moment that the head of the convoy leaves r until the moment that the tail arrives in r again (making the same assumptions when leaving and entering r as above for s and t). Note that both problems can also be defined for undirected graphs, by simply replacing every undirected edge by two directed edges for both directions, with the same length and speed. Finally, in Convoy-CPP we assume that our graph G is undirected (as it is common when studying the Chinese postman problem in the classical setting, e.g., [3]) but again each edge $e \in E$ has a length $\ell(e) \geq 0$ and a speed $q(e) > 0$. Also, like in Convoy-TSP we are given a special vertex $r \in V$ in which the convoy starts and ends. The goal is to find a tour that traverses every edge at least once, minimizing the time between the moment that the head of the convoy leaves r until the moment that the tail arrives in r again. In all problems, we assume without loss of generality that $\min_{e \in E} q(e) = 1$.

Convoy-SPP is the most basic routing problem for a convoy and it is motivated by any movement of a convoy from one point to another in a street network. The problem was introduced in [8] where the authors give an exponential time exact algorithm and argue that this "hints" that the problem is NP-hard. The Convoy-TSP problem arises for example when relief supplies need to be delivered to several locations, e.g., to several villages or cities, after a natural disaster or in unsecure territory. In these cases, moving in a convoy may be necessary so that the vehicles can support each other when roads are interrupted, or for protection. Similarly, Convoy-CPP arises when instead the convoy needs to visit all edges, e.g., because several villages are located on roads connecting the major cities. We are not aware of prior results for Convoy-TSP or Convoy-CPP. The setting of a convoy moving through a network can be considered as a variant of dynamic flows or flows over time, see [12].

1.1 Our contributions

We give an exact algorithm for Convoy-SPP that runs in strongly polynomial time. Note that this contrasts the argumentation mentioned above in [8]. A natural approach for Convoy-SPP would be to generalize algorithms for the usual shortest path problem, like Dijkstra's algorithm or Bellman-Ford. However, it is not clear how to do this. In both algorithms, we compute a value for each vertex v that indicates the distance of a shortest path from s to v (in the case of Bellman-Ford, we have one such value for each possible number of edges on this path). However, for continuing this path from v to t, it is crucial to know the speeds of the edges in which the convoy is located when reaching v. For example, there can be a long path from s to v such that the whole convoy is located on very fast edges when reaching v, and a short path such that the whole convoy is located on very slow edges when reaching v, and many other Pareto-optimal possibilities in between. We cannot afford to compute each of these possibilities. Instead, we use a very different approach. We guess the slowest edge (u, v) on the optimal path from s to t. The remaining problem splits into two independent subproblems: finding the optimal path from s to u and finding the optimal path from v to t. For each of these subproblems, we can assume that no edge slower than (u, v) is used. This leads to a polynomial number of possible subproblems which we solve via dynamic programming. Here, let n and m denote the number of vertices and edges in the graph, respectively.

▶ **Theorem 1.** *There is an algorithm for the Convoy-SPP with a running time of* $O(nm^2)$.

For usual TSP and asymmetric TSP there are polynomial time approximation algorithms known, e.g., [6, 14, 15]. However, it is not clear how to generalize them to Convoy-TSP, even if the graph were undirected. A natural attempt would be to define the cost of an edge e to be $\ell(e)/q(e)$, i.e., the time that it takes for an infinitesimally small convoy to traverse only this edge. However, the actual convoy of length L might be much slower than this on e if a part of it is located on a slower edge. In particular, even one single edge e can slow down the convoy for $L/q(e)$ time units which might be much larger than $\ell(e)/q(e)$. On the other hand, we cannot add this value $L/q(e)$ as a penalty to the cost of e, since if the convoy traverses many edges of speed $q(e)$, this penalty would overcount the slowdown by an arbitrarily large amount. Also, it is not obvious how to compute good lower bounds on the optimal solution: the strongest lower bound used in the known results for (asymmetric) TSP is the Held-Karp LP-relaxation [5] which does not take the convoy feature into account. It is not clear how to add this aspect to the LP.

Instead, we use a reduction to group-TSP (in the usual setting, not in the convoy setting). The intuition is the following. Suppose that $v_1, v_2, ..., v_n$ is the order in which the vertices are visited for the first time in the optimal solution. We identify that a key question is to determine for each vertex v_j the minimum speed at which the convoy travels while going from v_{j-1} to v_j. Therefore, in our reduction to group-TSP we introduce a copy of v_j for each possible speed and assign all these copies into the same group, i.e., only one of these copies needs to be visited. We introduce edges between these copies, corresponding to paths that use only edges with the respective minimum speeds, using our algorithm for Convoy-SPP as a subroutine. We show that our reduction loses only a constant factor in the approximation guarantee. By scaling the speeds to powers of 2, we ensure that there are intuitively only $O(\log n)$ different speeds, and thus each group for some vertex v_j has at most $O(\log n)$ copies of v_j. For such instances of group-TSP there is an $O(\log n)$-approximation [13] which we invoke as a subroutine.

We give a simple reduction from Convoy-CPP to Convoy-TSP which yields an $O(\log n)$-approximation algorithm for Convoy-CPP as well.

▶ **Theorem 2.** *There is a polynomial time $O(\log n)$-approximation algorithm for Convoy-TSP.*
▶ **Theorem 3.** *There is a polynomial time $O(\log n)$-approximation algorithm for Convoy-CPP.*

Maybe surprisingly, we prove that Convoy-CPP is NP-hard, which contrasts the fact that the usual Chinese postman problem is solvable in polynomial time [3].

▶ **Theorem 4.** *The Convoy-CPP is* NP*-hard, both for directed and undirected graphs, even when all edge lengths are equal to 1 and $L = 2$.*

Finally, we give an $O(1)$-approximation algorithm for Convoy-TSP in undirected trees. While for normal TSP in trees, a simple depth-first-search (DFS) is optimal, this is *not* the case for Convoy-TSP. In fact, DFS can yield a very bad solution, already on stars in which the edges have all length 1 and only two different speeds: assume that r is the center of the star, then the optimal solution visits first all slow edges and then all fast edges (or vice versa) while in a DFS solution this order can be arbitrary and the convoy could travel all the time with the low speed, which can be much slower than the optimal solution. While this particular example has an obvious optimal solution, it is not clear how to generalize it to arbitrary trees. Instead, we use a recursive approach. It is useful to imagine that our goal is to solve Convoy-CPP, which is equivalent to Convoy-TSP on trees, and hence we want to compute a tour that traverses all edges. We round the speeds to powers of 2 and form clusters of the edges with the lowest speed such that, intuitively, the edges in each cluster are close together and any two different clusters are far apart (at least L units). For each cluster we compute a DFS tour and argue that the optimal solution cannot be much faster when it traverses the edges of the cluster. Then, we remove all edges of the lowest speed and recurse on each connected component.

▶ **Theorem 5.** *There is a polynomial time $O(1)$-approximation algorithm for Convoy-TSP and Convoy-CPP on undirected trees.*

We defer the proof of this theorem to the full version of the paper. Our $O(1)$-approximation for trees is non-trivial and, although the analysis leaves room for a modest improvement of the constant 13, getting a ratio close to 1 needs a substantially different approach. In fact, an exact algorithm may still be possible. Also, it remains an open problem whether an $O(1)$-approximation for Convoy-TSP in arbitrary graphs exists.

1.2 Related work

The authors of [8] actually refer to our Convoy-SPP as the convoy quickest path problem. This is because there are some similarities between Convoy-SPP and the quickest path problem (QPP) [7]. In the QPP, we are given a directed graph $G = (V, E)$, where each edge $e \in E$ has a time value $d(e) \geq 0$, and a capacity $c(e) \geq 0$. The capacity is an upper bound on the number of units per time unit that can pass through an edge. Here, we are also given two special vertices $s, t \in V$, and a length of a transmission $\sigma \geq 0$. The goal in QPP is to find a path from s to t minimizing the total time spent on the path plus the delay caused by the edge capacities. Note that this is different from our model in which the edges have speeds. Several polynomial time exact algorithms have been developed for QPP, we refer to [10] for a survey.

Another related problem is the convoy movement problem as studied in [1]. It is concerned with routing *multiple* convoys simultaneously such that there is no moment in time in which two convoys cross paths. Also in the variable speed convoy movement problem, studied in [11], the goal is to route several convoys to their respective destinations such that their paths do not cross.

1.3 Outline

The rest of the paper is organized as follows. In Section 2 we will present a dynamic programming algorithm that solves Convoy-SPP to optimality in polynomial time. Then in Section 3 we consider Convoy-TSP on arbitrary directed graphs, and we give an $O(\log n)$-approximation, where n is the number of vertices. In Section 4, we will show that Convoy-CPP is NP-hard, and show how the results for Convoy-TSP can be used to obtain similar results for Convoy-CPP.

2 Convoy-Shortest Path Problem

In this section we present an algorithm for Convoy-SPP with a running time of $O(nm^2)$, i.e., we prove Theorem 1. Without loss of generality we assume that the edges are indexed in non-increasing order of their speed, i.e., such that $q(e_1) \geq q(e_2) \geq \ldots \geq q(e_m) = 1$. First, we show that we can restrict ourselves to instances of the form described in the statement of the following lemma. This will simplify our algorithm since it eliminates certain special cases, e.g., when the distance between s and t is shorter than L.

▶ **Lemma 6.** *Without loss of generality we can assume that s has exactly one outgoing edge e with $q(e) = 1$ and $\ell(e) = L$, and t has exactly one incoming edge e' with $q(e') = 1$ and $\ell(e') = L$.*

Proof. Consider an instance I of the convoy-shortest path problem on a graph $G = (V, E)$ with speeds $q(e_1) \geq \ldots \geq q(e_m) = 1$. Create an instance I' on the graph $G' = (V', E')$ with $V' = V \cup \{s', s'', t', t''\}$, where E' includes all edges of E, but all edges connected to s and t are now connected to s'' and t'' respectively. We also add the new edges (s, s'), (s', s''), (t'', t') and (t', t) to E' with speeds $q((s, s')) = q((t', t)) = 1$ and $q((s', s'')) = q((t'', t')) = q(e_1)$, all of length L.

Any path in I can be transformed into a corresponding path in I' in which the convoy first traverses the edges (s, s') and (s', s''), then takes the corresponding edge from s'' followed by the same edges as in I, and finishes its path with the corresponding edge to t'' and finally edges (t'', t') and (t', t). The time of the path in I' is exactly $4L$ longer than the time of the path in I. We can transform any path in I' similarly into a corresponding path in I that is $4L$ shorter.

Since the solution values in I and I' differ only by a global constant that is independent of the respective solution, we can make this transformation for any instance I without loss of generality, solve the solution on the modified instance I', and transform its solution back to a solution for I. ◀

From now on we denote with I' a given instance that satisfies the properties of Lemma 6. Our algorithm is a dynamic program (DP) that uses a divide and conquer strategy. Intuitively, we first guess the edge $e_i = (v, w)$ with the largest index i on the path of the optimal solution; hence, no edge is slower on this path. This splits this path into three parts: a path from s to v, the edge (v, w), and a path from w to t. Note that the first and the last part use only edges that have a smaller index than i, i.e., that are at least as fast as (v, w). We recurse in order to find these two subpaths, i.e., we will ensure that there is a DP-cell for each of these subproblems. These two subproblems are independent, since while a part of the convoy is located on the edge (v, w), that edge defines the speed of the convoy.

Formally, our DP-table has a cell $CSP(u, (v, w))$ for combinations of a vertex $u \in V$ and an edge $e_i = (v, w) \in E$. This cell encodes the following subproblem. First, place the convoy in the graph such that its tail is located on u, in any way you like. Then, we need to find a

path such that beginning with this starting position, the head of the convoy reaches v as fast as possible, while we are only allowed to use edges in the set $E_{(v,w)} = \{e_1, \ldots, e_{i-1}\}$ (i.e., edges with index lower than the index of (v, w) and which are hence at least as fast as (v, w)). In this cell $CSP(u, (v, w))$ we store the time the convoy needs to traverse this path; we also store the path itself in order to determine later the optimal path and not only its time. Note that the edges on which we place the convoy in the first step are part of the computed path but there is no cost involved in this placement. Also note that it is sufficient that the head of the convoy reaches v but it does not need to traverse the edge (v, w). We include (v, w) in the description of the cell $CSP(u, (v, w))$ because it determines $E_{(v,w)}$, the subset of the edges that the convoy is allowed to use in its path. We introduce a cell $CSP(u, (v, w))$ for each combination of a vertex $u \in V$ and an edge $e_i = (v, w) \in E$ such that the shortest path from u to v using only edges in $E_{(v,w)}$ has a length of more than L. We will see later that if the shortest path from u to v has length at most L, then we do not need to recurse on that subproblem but we can simply take the shortest path from u to v in $E_{(v,w)}$.

Suppose that we are given a $CSP(u, (v, w))$. Let $e_{i'} = (v', w') \in E_{(v,w)}$ be the edge with largest index i' in the path P^* of the optimal solution corresponding to the cell $CSP(u, (v, w))$. Suppose that the part of P^* between u and v' has a length of at least L and that the same holds for the part of P^* between w' and v. Then the former part is the optimal solution to the DP-cell $CSP(u, (v', w'))$ and after that the convoy needs $(\ell(e_{i'}) + L)/q(e_{i'})$ time units for traversing $e_{i'}$. The last part of P^* would be the optimal solution to the DP-cell $CSP(v, (w', v'))$ if the edges of G were all reversed, so if each edge $(\hat{u}, \hat{v})$ were replaced by the edge $(\hat{v}, \hat{u})$ with the same length and speed as $(\hat{u}, \hat{v})$. Therefore, we define a "reversed" auxiliary graph $G^R = (V, E^R)$ with $E^R = \{(v, u) \mid (u, v) \in E\}$ where every reversed edge $e^R \in E^R$ has the same speed and length as its original $e \in E$. Also, we define the same DP-cells as above for the same subproblems for G^R, i.e., we define a cell $CSP^R(\hat{u}, (\hat{w}, \hat{v}))$ for each combination of a vertex $\hat{u} \in V$ and an edge $(\hat{w}, \hat{v}) \in E^R$ (so $(\hat{v}, \hat{w}) \in E$) such that the shortest path from $\hat{u}$ to $\hat{w}$ in G^R has a length of more than L. Thus, we can split P^* into the optimal solution for $CSP(u, (v', w'))$, the edge (v', w'), and the optimal solution to $CSP^R(v, (w', v'))$, and it takes the convoy a total time of $CSP(u, (v', w')) + (\ell(e_{i'}) + L)/q(e_{i'}) + CSP^R(v, (w', v'))$ to traverse P^*.

If the part of P^* from u to v' has length at most L, then one can show that it coincides simply with the shortest path from u to v using only edges in $E_{(v',w')}$ (since the edge (v', w') determines the speed of the convoy anyway), and a similar statement holds for the part of P^* from w' to v. Thus, when we want to compute the entry of the cell $CSP(u, (v, w))$, we guess the edge $e_{i'} = (v', w')$, i.e., we try each edge in $E_{(v,w)}$, we compute the lengths of the shortest paths from u to v' and from w' to v using only edges in $E_{(v',w')}$, and depending on their lengths look up the DP-cells $CSP(u, (v', w'))$ and $CSP^R(v, (w', v'))$ or take these shortest paths directly.

This yields the following recursive formula, where for each combination of a vertex u and an edge $(v, w) \in E$ we denote by $SP(u, (v, w))$ the shortest path in G from u to v that uses only edges in $E_{(v,w)}$; we define $SP^R(u, (v, w))$ similarly for G^R.

$$CSP(u, (v, w)) = \min \left\{ \begin{array}{l} \min\limits_{\substack{e \in E_{(v,w)}: \\ SP(u,e) > L, \\ SP^R(v,e^R) > L}} \left\{ CSP(u, e) + \dfrac{\ell(e) + L}{q(e)} + CSP^R(v, e^R) \right\}, \\ \min\limits_{\substack{e \in E_{(v,w)}: \\ SP(u,e) \leq L, \\ SP^R(v,e^R) > L}} \left\{ \dfrac{SP(u, e) + \ell(e)}{q(e)} + CSP^R(v, e^R) \right\}, \end{array} \right. \tag{1}$$

$$\min_{\substack{e\in E_{(v,w)}:\\ SP(u,e)>L,\\ SP^R(v,e^R)\leq L}} \left\{ CSP(u,e)) + \frac{\ell(e) + SP^R(v,e^R)}{q(e)} \right\},$$

$$\min_{\substack{e\in E_{(v,w)}:\\ SP(u,e)\leq L,\\ SP^R(v,e^R)\leq L}} \left\{ \frac{SP(u,e) + \ell(e) + SP^R(v,e^R) - L}{q(e)} \right\} \Bigg\}.$$

Note that we need not define base cases for this dynamic program. All cells where both subpaths are shorter than L are computed using SP. In the first cell where we recurse on a previously computed cell C of the DP, the length of the path corresponding to C is larger than L, but cell C has already been computed at that point because its value will be attained because of the fourth term of the recursion using SP.

We could compute all shortest paths in the graph restricted to $E_{(v,w)}$ for each $(v,w) \in E$ with any shortest path algorithm, e.g., with Dijkstra's algorithm. However, in order to improve the running time slightly, we use another DP which is analogous to the DP above. We initialize $SP(v,(v,w)) = 0$ for all $(v,w) \in E$, $SP^R(v,(v,w)) = 0$ for all $(v,w) \in E^R$, and we use the recurrence

$$SP(u,(v,w)) = \min_{e\in E_{(v,w)}} \left\{ SP(u,e) + \ell(e) + SP^R(v,e^R) \right\},$$

for each cell $SP(u,(v,w))$ with $u \neq v$ and a symmetric recurrence for each entry $SP^R(u,(v,w))$ for which $u \neq v$.

Finally, we output the path stored in $CSP(s,(t',t))$. Note that the time stored in that cell corresponds to the time that the convoy, starting with its tail in s, needs such that its head reaches t'. The convoy needs an additional L time units in order to leave s, an additional L time units to reach t, and an additional L time units to enter t completely. This yields our solution to the transformed instance I', which we can transform back to a solution to the original instance I using Lemma 6.

▶ **Theorem 1.** *There is an algorithm for the Convoy-SPP with a running time of $O(nm^2)$.*

Proof. First we prove correctness and then we prove the running time.

Consider a cell $CSP(u,(v,w))$ for which we want to compute the optimal solution, which is a path P from u to v that minimizes the convoy travel time. We guess the slowest edge in P, let this be $e^* = (u^*,v^*)$. Let P_1 and P_2 be the subpaths of P before and after e^* respectively (both excluding e^*).

First consider the case where both P_1 and P_2 are strictly larger than L. Then we can subdivide the problem encoded in this cell of the DP by using edge e^*. The total time it takes for the convoy to travel along P is the time to travel along P_1 plus the time to travel along e^* plus the time to travel along P_2. Since both P_1 and P_2 are larger than L and they do not contain this new bottleneck edge e^*, their travel time is equal to $CSP(u,e^*)$ and $CSP^R(v,e^*)$ respectively. Note that these terms indeed do not contain the travel time along e^*, as the first term is the travel time until the head of the convoy reaches e^* and the second term is the travel time from the point in time onward when the tail leaves e^*. Finally, the travel time along e^* that is missing is the time it takes for the convoy to completely traverse the edge e^* from head to tail. This equals $(\ell(e^*) + L)/q(e^*)$, since this bottleneck edge determines the speed of the convoy for all these time units. Together, this gives us the first term of Equation (1).

Now observe that if $SP(u, e^*) \leq L$, there exists a path from u to u^* such that the tail of the convoy has just left u and some part of the convoy has reached u^*. Thus, edge e^* is the edge that determines the speed of the convoy while it is moving along P_1 and along e^*. Since it travels a distance of $SP(u, e^*) + \ell(e^*)$ there at speed $q(e^*)$, this takes $(SP(u, e^*) + \ell(e^*))/q(e^*)$ time units. Then, if the second part of the path is longer than L (i.e., if $SP^R(v, e^*) > L$), P_2 is the path corresponding to the solution $CSP^R(v, e^*)$ and we retrieve the second term of Equation (1).

Similarly, if $SP^R(v, e^*) \leq L$, then $P_2 = SP^R(v, e^*)$, and by the same arguments as above we see that the third term of Equation (1) is correct.

Finally, if both $SP(u, e^*) \leq L$ and $SP^R(v, e^*) \leq L$, then this total path is traversed at speed e^*. The total length of the path that the convoy needs to cover between u and v is therefore $SP(u, e^*) + \ell(e^*) + SP^R(v, e^*)$. Since we measure the time it takes for the tail to leave u and the head to reach v, we need to subtract L from this distance to get the distance that the convoy traverses. Note that the distance $SP(u, e^*) + \ell(e^*) + SP^R(v, e^*) - L$ must be positive because we only introduce this cell whenever $SP(u, (v, w)) > L$. Dividing by the speed $q(e^*)$ at which the convoy traverses this distance, we obtain the last term of Equation (1). This finishes the proof of correctness of the algorithm.

Using these observations, the entry of the cell $CSP(u, (v, w))$ can be computed using only previously computed entries of cells of shorter paths. There are $O(nm)$ cells in each of the dynamic programs. In each computation there are $O(m)$ terms that need to be compared for the minimum value, since we recurse on one of the edges of the path. Therefore, the total running time of each of the dynamic programs is bounded by $O(nm^2)$. ◀

3 Convoy-TSP in general graphs

In this section we present our $O(\log n)$-approximation algorithm for Convoy-TSP in general graphs. First, we do some guesses and preprocessing in order to simplify our problem, while losing only a constant factor in our approximation guarantee.

▶ **Lemma 7.** *By losing a factor of 2 in the approximation ratio, we can assume that $\ell(e) \geq L/nm$ for each edge e.*

Proof. Let $\epsilon = L/nm$. First, increase the length of any edge with length smaller than ϵ to ϵ. It follows that the optimal value of the modified instance cannot be smaller than the optimal value of the original instance. Now, consider the optimal solution of the original instance. Let e^* be the slowest edge with length less than ϵ that is traversed by the optimal solution. We know that the optimal value of the original instance is at least $L/q(e^*)$. If we follow the same tour in the modified instance, we have to spend at most $\epsilon/q(e^*)$ time additionally for each modified edge. This is true since staying longer on an edge will not decrease the speed of the convoy. In total, we spend at most $nm\epsilon/q(e^*) = L/q(e^*)$ time units extra, so we lose at most a factor 2. ◀

From now on we assume that all edges have length at least L/nm. We say that an edge e is *short* if $\ell(e) \leq 2L$ and *long* otherwise. Intuitively, the long edges are easy to deal with: if the convoy traverses a long edge e, then the whole convoy will be on e while traversing a distance of at least L, which takes at least $L/q(e)$ time. Potentially, for some time only a part of the convoy is on e and then e slows the convoy down. However, this time is at most $2L/q(e)$ which can be charged to the previous time of $L/q(e)$. In particular, if we had only long edges then we could reduce the problem to (normal) TSP by losing only a factor $O(1)$ by simply giving each (long) edge length $(\ell(e) + L)/q(e)$.

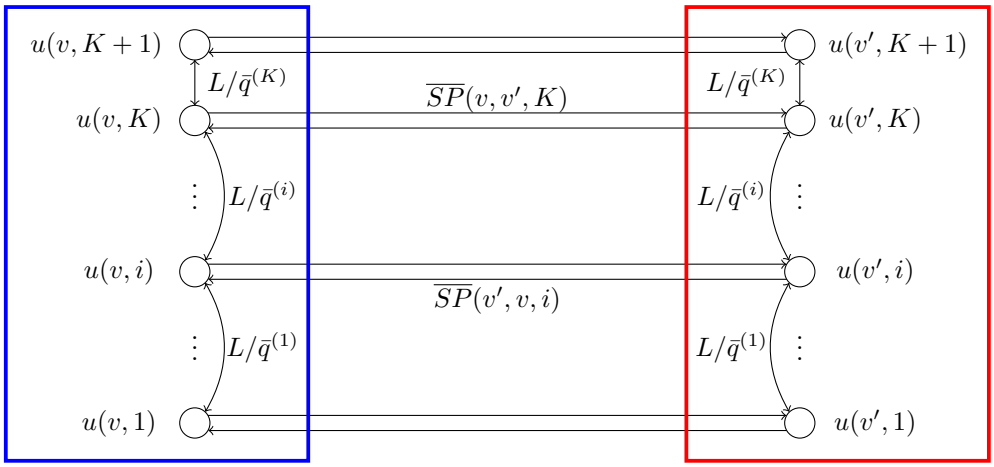

Figure 2 Illustration of constructed instance of group-TSP. Note that there is an edge between any pair of the form $u(v, i)$ and $u(v, j)$. From the set of vertices enclosed by both the blue and red rectangle, at least one vertex has to be visited.

Thus, the core difficulty of the problem stems from the short edges. We want to modify the instance such that the short edges have only $O(\log n)$ different speeds.

▶ **Lemma 8.** *By losing a factor of 6 in the approximation ratio, we will assume that the short edges have only $O(\log n)$ different speeds and that they are all powers of 2.*

Proof. We guess the short edge with the lowest speed that is used in OPT. Let e^* be this edge. First, we remove each short edge that is slower than e^*. Then, if a short edge e is faster than $n^2m^2 \cdot q(e^*)$, we round its speed down to $n^2m^2 \cdot q(e^*)$. Since these edges are short, and any tour uses at most nm edges, this rounding will incur at most an additional

$$nm \cdot \frac{2L}{n^2m^2q(e^*)} = \frac{2L}{nmq(e^*)} \leq \frac{2\ell(e^*)}{q(e^*)} \leq 2\text{OPT}$$

time units, where the first inequality follows from Lemma 7. Hence, we lose at most a factor 3 in the approximation ratio. Finally, we round all speeds of the short edges down to powers of 2, which loses us another factor 2, to obtain $O(\log n^2m^2) = O(\log n)$ different speeds for the short edges. ◀

Let $\bar{q}^{(1)} < \bar{q}^{(2)} < \ldots < \bar{q}^{(K)}$ denote the different speeds of the short edges, and by Lemma 8 we have that $K = O(\log n)$. We guess the speed of the slowest short edge used in OPT and remove all slower short edges. Hence, we assume w.l.o.g. that OPT uses a short edge of speed $\bar{q}^{(1)}$.

We define now an instance of asymmetric group-TSP to which we reduce our given instance of Convoy-TSP. In group-TSP, the input consists of a directed graph $G' = (V', E')$, a weight $w(e)$ for each $e \in E'$ and a set of of groups $\mathcal{V}'_1, ..., \mathcal{V}'_s \subseteq V'$. Intuitively, the edge weights denote the lengths of the edges E', but we call them weights here in order to distinguish them better from the lengths $\ell(e)$ of the edges $e \in E$. The goal is to compute a minimum weight tour that visits at least one vertex from each group $\mathcal{V}'_i$.

For each input vertex $v \in V$ we define a group of $K+1$ vertices $\{u(v, 1), ..., u(v, K+1)\}$ (see Figure 2), i.e., the tour needs to visit at least one of these vertices. Intuitively, for each $i \in \{1, ..., K\}$ the vertex $u(v, i)$ corresponds to arriving at the vertex v at speed $\bar{q}^{(i)}$, while the vertex $u(v, K+1)$ corresponds to arriving at v after traversing a long edge e (and then

the speed does not matter much since any slowdown due to e can be charged to the time the convoy spends on e). For any pair of vertices $v, v' \in V$ and each $i \in \{1, ..., K+1\}$ we introduce a directed edge $e = (u(v,i), u(v',i))$ and we define the weight $w(e)$ to be the time of the quickest path from v to v' such that

- at the beginning the head of the convoy is on v and the convoy travels at speed at most $\bar{q}^{(i)}$ until the tail of the convoy has left v,
- at the end the head of the convoy is at v',
- the convoy is allowed to use only short edges of speed at least $\bar{q}^{(i)}$ and long edges (of any speed); if $i = K+1$ it is only allowed to use long edges.

We denote by $\overline{SP}(v, v', i)$ the corresponding path. We can compute $\overline{SP}(v, v', i)$ with our algorithm from Section 2.

▶ **Lemma 9.** *For each $v, v' \in V$ and each $i \in \{1, ..., K+1\}$ we can compute $\overline{SP}(v, v', i)$ in time $O(nm^2)$.*

Proof. From graph G, create an auxiliary graph $\bar{G}$ with a new vertex $\bar{v}$ and an edge $\bar{e} = (\bar{v}, v)$ with speed $\bar{q}^{(i)}$ and length L. Moreover, delete all short edges from E' with a speed slower than $\bar{q}^{(i)}$. Indeed, the solution to that instance gives us exactly $\overline{SP}(v, v', i)$ as requested, which can be solved using the algorithm from Theorem 1. ◀

Also, for each vertex v and any $i, i' \in \{1, ..., K+1\}$ we introduce the edge $e = (u(v,i), u(v,i'))$, and define its weight $w(e) := L/\min\{\bar{q}^{(i)}, \bar{q}^{(i')}\}$. Intuitively, this models that if the convoy traverses some edge of speed $\bar{q}^{(i)}$ and after that edges of a higher speed $\bar{q}^{(i')}$, then the edge of speed $\bar{q}^{(i)}$ slows down the convoy for $L/\bar{q}^{(i)}$ more time units. To streamline our argumentation, we introduce this edge with this cost also if $\bar{q}^{(i')} < \bar{q}^{(i)}$.

We solve our instance of group-TSP with an adaptation of the algorithm in [13] which yields an $O(K)$-approximate tour T'. We translate T' to a tour T of our given instance of Convoy-TSP as follows. We define T by adding to it step by step the edges that the convoy needs to traverse. We know that T' must visit at least one vertex $u(r,i)$ of the group corresponding to the initial vertex $r \in V$ of the convoy. Starting in $u(r,i)$, we follow T'. Whenever T' uses an edge $e = (u(v,i), u(v',i))$ for two vertices v, v' and an $i \in \{1, ..., K+1\}$ then we add to T the edges in $\overline{SP}(v, v', i)$. When the convoy travels along an edge $e = (u(v,i), u(v,i'))$ for a vertex v and $i, i' \in \{1, ..., K+1\}$ we do not add edges to T (such an edge e ensures only that the time of our convoy in tour T is comparable to $w(T')$).

▶ **Lemma 10.** *The convoy needs time $O(w(T'))$ to traverse the tour T.*

Proof. Suppose that when the convoy traverses T it does this during the time interval I. Let I_1 denote the union of all subintervals of I during which the speed of the convoy is determined by a long edge e. Consider one such edge e. Then there are two vertices v, v' and an $i \in \{1, ..., K+1\}$ such that $(u(v,i), u(v',i)) \in T'$ and $e \in \overline{SP}(v, v', i)$. Thus, the time during I in which e determines the speed of the convoy is at most $\frac{L+\ell(e)}{q(e)}$, while e contributes at least $\frac{\ell(e)}{q(e)} = \Omega\left(\frac{L+\ell(e)}{q(e)}\right)$ to $w((u(v,i), u(v',i)))$. Therefore, $|I_1| \in O(w(T'))$.

Let $I_2 := I \setminus I_1$, i.e., the union of the time intervals in I during which the speed of the convoy is determined by a short edge. We want to bound $|I_2|$. To this end, take T' and consider all connected components of edges of the form $e = (u(v,i), u(v',i))$ for two vertices v, v' and an $i \in \{1, ..., K+1\}$. Let $C = \{e_1, e_2, ...\}$ be such a component. Then there exists an $i \in \{1, ..., K+1\}$ such that for each edge $e_j \in C$ there are two vertices $v_j, v'_j \in V$ such that $e_j = (u(v_j, i), u(v'_j, i))$ and the convoy traverses the path $\overline{SP}(v_j, v'_j, i)$. The total time that the convoy traverses the paths $\left\{\overline{SP}(v_j, v'_j, i)\right\}_{j:e_j \in C}$ such that the speed of the convoy

is determined by a short edge of speed *at least* $\bar{q}^{(i)}$ is bounded by $\sum_{j:e_j \in C} w(e_j)$ (note that $w(e_j)$ might over-estimate the time needed by the convoy to traverse the edges of $\overline{SP}(v_j, v'_j, i)$ since we assume that the convoy travels at speed at most $\bar{q}^{(i)}$ when leaving v_j). It remains to bound the time that the convoy traverses the paths $\left\{\overline{SP}(v_j, v'_j, i)\right\}_{j:e_j \in C}$ such that the speed of the convoy is determined by a short edge of some speed that is strictly *slower* than $\bar{q}^{(i)}$; we denote by $\hat{t}$ this amount of time. Let $\bar{q}^{(i^*)}$ be the smallest such speed. In particular, this implies that $\hat{t} \leq L/\bar{q}^{(i^*)}$.

Let C' denote the connected component of the edges of the form $e = (u(v,i), u(v,i'))$ for a vertex $v \in V$ and $i, i' \in \{1, ..., K+1\}$ that is traversed in T' right before C. If $\hat{t} > 0$, then C' must contain an edge $e^* = (u(v,i^*), u(v,\hat{i}))$ with $\bar{q}^{(i^*)} < \bar{q}^{(\hat{i})}$ and thus $w(e^*) = L/\bar{q}^{(i^*)} \geq \hat{t}$. Hence, the total time that the convoy traverses C is bounded by $w(e^*) + \sum_{j:e_j \in C} w(e_j)$. Applying this argumentation to all components C yields that $|I_2| \in O(w(T'))$ and thus $|I| = |I_1| + |I_2| \ \in O(w(T'))$. ◀

Abusing notation, we denote by OPT the time that it takes to traverse OPT. It remains to show that $w(T') = O(\log n \cdot \mathrm{OPT})$. To this end, we show that we can construct a tour T^* for our instance of group-TSP for which $w(T^*) = O(\mathrm{OPT})$ holds. This implies that $w(T') = O(K \cdot w(T^*)) = O(\log n \cdot \mathrm{OPT})$. Note that hence our reduction to group-TSP loses only a factor $O(1)$ (and we lose the factor $O(\log n)$ because we can solve the instance of group-TSP only approximately).

Based on OPT, we describe now how we construct T^* such that $w(T^*) = O(\mathrm{OPT})$. Assume w.l.o.g. that $V = \{v_1, ..., v_n\}$ such that $v_1 = r$ (i.e., the initial vertex) and for each $j \in \{1, ..., n-1\}$ we have that v_j is visited in OPT for the first time before v_{j+1} is visited in OPT for the first time (note that OPT might visit a vertex several times). For convenience, we define that $v_{n+1} = v_1$. For each $j \in \{1, ..., n\}$ we do the following: let P_j denote the path that the convoy takes between visiting v_j and v_{j+1} for the first time. If P_j contains a short edge, we add $(u(v_j, i), u(v_{j+1}, i))$ to T where $\bar{q}^{(i)}$ is the minimum speed at which the convoy travels in OPT while the head of the convoy is on an edge in P_j. If P_j does not contain a short edge, then we add $(u(v_j, K+1), u(v_{j+1}, K+1))$ to T.

It can happen that for some vertex v_j we added two edges $(u(v_{j-1}, i), u(v_j, i))$, $(u(v_j, i'), u(v_{j+1}, i'))$ with $i \neq i'$. In this case we add the edge $(u(v_j, i), u(v_j, i'))$. Let T^* be the resulting tour.

▶ **Lemma 11.** *The group-TSP solution T^* has total weight $w(T^*) = O(\mathrm{OPT})$.*

Proof. Suppose that we change OPT to a new tour OPT′ by slowing down the convoy even though it would not be necessary. We do this such that OPT′ can be traversed in at most time $O(\mathrm{OPT})$. Let $\ell(\mathrm{OPT})$ denote the total length that the head of the convoy travels in OPT. Here, we assume w.l.o.g. that the head keeps traveling after reaching r for a distance of L using edges of unbounded speed. Now, consider the interval $I = [0, \ell(\mathrm{OPT}))$. For each point $d \in I$ denote by $q_{\mathrm{OPT}}(d)$ the speed that the convoy travels in OPT when the head of the convoy has already traveled a distance of d. Let $I' = [a, b) \subseteq I$ denote a maximally large subinterval of I such that $q_{\mathrm{OPT}}(d) = \bar{q}^{(1)}$ for each $d \in I'$. Let a' denote the largest value $x \leq a$ such that after travelling a distance of x in OPT the head of the convoy visits a vertex $v \in V$ for the first time. We define a new speed function q' for the head of the convoy for which we define for each $d \in [a', \min\{a' + L, b\})$ that $q'(d) := \bar{q}^{(1)}$.

Similarly, let b' denote the smallest value $y \geq b$ such that after travelling a distance of y in OPT the head of the convoy visits a vertex $v \in V$ for the first time. We define q' such that for each $d \in [\max\{b' - L, a\}, b')$ we define the speed $q'(d) := \bar{q}^{(1)}$ for the head of the convoy. We define that the interval $[a', b')$ is *clean.* We do this operation for each interval

$I' \subseteq I$ such that $q_{\mathrm{OPT}}(d) = \bar{q}^{(1)}$ for each $d \in I'$. Let I_2 denote the union of subintervals of I that we obtain if we remove from I each clean interval $I' = [a', b')$. Inductively, we do the same transformation for each maximally large interval I' of I_2 such that $q_{\mathrm{OPT}}(d) = \bar{q}^{(2)}$ for each $d \in I'$, obtaining a set I_3. We continue for each $k' \in \{3, 4, ..., K\}$.

If for a value $d \in [0, \ell(\mathrm{OPT}))$ we did not define a value $q'(d)$, then we define $q'(d) := q_{\mathrm{OPT}}(d)$. Let OPT' denote a tour in which we traverse the edges of OPT at the speeds given by q'. Observe that $q'(d) \leq q_{\mathrm{OPT}}(d)$ for each $d \in [0, \ell(\mathrm{OPT}))$. Using that the speeds are powers of 2 and geometric sum arguments, one can show that $\mathrm{OPT}' \in O(\mathrm{OPT})$, see Lemma 13 in the appendix.

We want to argue that $w(T^*) \in O(\mathrm{OPT}')$. Consider first the edges $e \in T^*$ of the form $e = (u(v, i), u(v', i))$ for two vertices v, v' and an $i \in \{1, ..., K+1\}$. Let T_1^* denote the set of all these edges and let $T_2^* := T^* \setminus T_1^*$. We first want to show that $w(T_1^*) \in O(\mathrm{OPT}')$. Let $C = \{e_1, e_2, ...\}$ be a connected component of T_1^*. Then there exists an $i \in \{1, ..., K+1\}$ such that for each edge $e_j \in C$ there are two vertices $v_j, v_j' \in V$ such that $e_j = (u(v_j, i), u(v_j', i))$. By definition of OPT' and $w(e_j)$, we have that $w(e_j)$ is at most the time that in OPT' the head of the convoy needs to move from v_j to v_j'. Repeating this argumentation for each edge $e_j \in T_1^*$, this implies that $w(T_1^*) \in O(\mathrm{OPT}')$.

It remains to argue that $w(T_2^*) \in O(\mathrm{OPT}')$. Assume that $V = \{v_1, ..., v_n\}$. For each vertex $v_j \in V$, denote by d_j the distance traveled by the head of the convoy when it reaches v_j for the first time. Assume w.l.o.g. that the vertices are ordered such that $d_j < d_{j'}$ for each j, j' with $j < j'$. This partitions the $[0, \ell(\mathrm{OPT}'))$ into intervals of the form $[d_j, d_{j+1})$ with $j \in \{1, ..., n\}$, where for convenience we define $d_{n+1} := \ell(\mathrm{OPT}')$. Consider an edge $e \in T_2^*$. By definition of T^*, there are vertices v_j, v_{j+1}, v_{j+2} and speeds $\bar{q}^{(i)}, \bar{q}^{(i')}$ such that $e = (u(v_{j+1}, i), u(v_{j+1}, i'))$ and at some point during the interval $[d_j, d_{j+1})$ or at some point during the interval $[d_{j+1}, d_{j+2})$ (measured in the total distance that the head of the convoy has traveled) the convoy travels at speed $\min\{\bar{q}^{(i)}, \bar{q}^{(i')}\}$ due to some edge e'. We charge $w(e) = L/\min\{\bar{q}^{(i)}, \bar{q}^{(i')}\}$ to the slow-down of the convoy due to e'. The remainder of the argumentation follows via geometric sum arguments, addressing the case when we charge two edges of different speeds that are traversed (partially) at the same time by the convoy in OPT'.

To this end, for each speed $\bar{q}^{(i)}$ we define $\mathcal{I}^{(i)}$ to be the family of maximal intervals I of interval $[0, \ell(\mathrm{OPT}'))$ during which the convoy travels at speed $\bar{q}^{(i)}$ or slower (again, measured in the total distance that the head of the convoy has traveled). For each speed $\bar{q}^{(i)}$ we define $|\mathcal{I}^{(i)}|$ to be the total length of the intervals in $\mathcal{I}^{(i)}$ and we observe that $|\mathcal{I}^{(i)}|/\bar{q}^{(i)} \leq \mathrm{OPT}'$. Since the speeds are powers of 2, via a geometric sum argument we also obtain that $\sum_i |\mathcal{I}^{(i)}|/\bar{q}^{(i)} \leq O(\mathrm{OPT}')$. We argue now how to charge $w(T_2^*)$ to $\sum_i |\mathcal{I}^{(i)}|/\bar{q}^{(i)}$. Consider an edge $e = (u(v_{j+1}, i), u(v_{j+1}, i')) \in T_2^*$ as defined above and corresponding speeds $\bar{q}^{(i)}, \bar{q}^{(i')}$. Let i^* be such that $\bar{q}^{(i^*)} = \min\{\bar{q}^{(i)}, \bar{q}^{(i')}\}$. Then there is an interval $I = [a, b) \in \mathcal{I}^{(i^*)}$ such that $a \in [d_j, d_{j+2})$ or $b \in [d_j, d_{j+2})$. We have that $w(e) \leq (b-a)/\bar{q}^{(i^*)}$. Thus, we charge $w(e)$ to I. In this way, each interval I in each set $\mathcal{I}^{(i)}$ is charged at most twice, and we obtain that hence the total charge of all edges in T_2^* amounts to $w(T_2^*) = \sum_{e \in T_2^*} w(e) \leq 2 \sum_i |\mathcal{I}^{(i)}|/\bar{q}^{(i)} \leq O(\mathrm{OPT}')$. Thus, $w(T^*) = w(T_1^*) + w(T_2^*) \in O(\mathrm{OPT}')$. ◀

We conclude that the time the convoy needs to traverse our computed tour T is bounded by $O(w(T')) = O(K \cdot w(T^*)) = O(\log n \cdot \mathrm{OPT})$. This yields our $O(\log n)$-approximation algorithm for the Convoy-TSP problem.

▶ **Theorem 2.** *There is a polynomial time $O(\log n)$-approximation algorithm for Convoy-TSP.*

4 Convoy-Chinese Postman Problem

In this section we discuss Convoy-CPP, which is defined on undirected graphs. First we show that Convoy-CPP is NP-hard, also for directed graphs. Note that CPP can be solved to optimality in polynomial time for both undirected [2] and directed graphs [3]. For mixed graphs, CPP is already NP-hard [9].

▶ **Theorem 4.** *The Convoy-CPP is* NP*-hard, both for directed and undirected graphs, even when all edge lengths are equal to 1 and* $L = 2$.

Proof. To show NP-hardness for undirected graphs we use a reduction from the Hamiltonian path problem [4]. In this problem we are given a graph $G = (V, E)$ and the question is whether there exists a Hamiltonian path, i.e., a path visiting all vertices exactly once, in G.

Let I be an instance of the Hamiltonian path problem. We create graph G' by copying the graph G of I. Each edge in G' has a length of 1 and a speed of $M = 4m + 1$, where m is the number of edges in G. We also pick an arbitrary vertex from this graph as the special vertex r. Now we create a new vertex in G' for every vertex in G, and connect it to its corresponding vertex. Each new edge has length 1 and speed 1. Further, we set L equal to 2. We will show that there is a solution in which the convoy needs less than $3n + 2$ time units to traverse all edges if and only if the original graph has a Hamiltonian path.

Suppose that G contains a Hamiltonian path. The convoy can first visit all edges with speed M spending at most $2m/M$ time units. It can then move to one of the endpoints of the Hamiltonian path, spending at most m/M time units. The convoy can now follow the Hamiltonian path, where it visits the edges with speed 1 upon arrival at a vertex. Since this will decrease the speed to 1, the head of the convoy will arrive at the other endpoint of the Hamiltonian path after $3n - 1$ more time units. While the head of the convoy heads back to r, the convoy will travel at speed 1 for $L = 2$ more time units. After that, the tail of the convoy still needs to travel a distance of at most m, while the convoy is at speed M. In total, the time spent by the convoy is at most

$$\frac{2m}{M} + \frac{m}{M} + 3n - 1 + L + \frac{m}{M} = \frac{4m}{M} + 3n - 1 + L < 3n + 2.$$

On the other hand, suppose that G does not contain a Hamiltonian path. In any solution, traversing the edges of speed 1 takes $2n$ time units. After the last visit to an edge with speed 1, the convoy moves at speed 1 for another $L = 2$ time units. The convoy also travels at speed 1 for at least one time unit between consecutive visits of speed 1 edges, since none of these edges are adjacent. Moreover, since G does not contain a Hamiltonian path, there needs to be at least one pair of speed 1 edges that are visited consecutively for which the convoy travels at speed 1 for at least two time units. In total, the time spent in any solution is at least $2n + n + L = 3n + 2$. To show NP-hardness for directed graphs we reduce from the directed Hamiltonian path problem [4] and perform a similar reduction. ◀

Next, we will relate the approximability of Convoy-CPP and Convoy-TSP. As a corollary, we also obtain an $O(\log n)$-approximation for Convoy-CPP on general graphs.

▶ **Lemma 12.** *For a given instance of Convoy-CPP whose optimal solution has value* α*, we can construct in polynomial time an instance of Convoy-TSP whose optimal solution has value* β *such that* $\beta \leq \alpha \leq 2\beta$.

Proof. First, replace each edge $(u, v) \in E$ with edges (u, w), (w, u), (w, v), and (v, w), where each of these edges gets length $\ell((u, v))/2$, and speed $q((u, v))$. The optimal solution to the given instance of Convoy-CPP is also feasible for the constructed instance of Convoy-TSP,

and thus $\beta \leq \alpha$. On the other hand, consider an optimal solution for the created Convoy-TSP instance with value β. If the resulting tour turns at w for some edge (u, v), we can adjust this tour to one that visits the original edge (u, v) fully at once, by moving back and forth between u and v when we visit w. Since this will not slow down the convoy, and the distance traveled along (u, v) is doubled, the total time spent will increase with at most a factor 2. This implies that $\alpha \leq 2\beta$. ◀

Together with our algorithm from Section 3, this yields our $O(\log n)$-approximation algorithm for Convoy-CPP.

▶ **Theorem 3.** *There is a polynomial time $O(\log n)$-approximation algorithm for Convoy-CPP.*

We remark that with similar arguments one can obtain an $O(\log n)$-approximation algorithm for Convoy-CPP on directed or on mixed graphs.

References

1 Pierre Chardaire, Geoff P. McKeown, S.A. Verity-Harrison, and S.B. Richardson. Solving a time-space network formulation for the convoy movement problem. *Operations research*, 53(2):219–230, 2005.

2 Jack Edmonds. The Chinese postman problem. *Operations Research*, 13:73–77, 1965.

3 Jack Edmonds and Ellis L. Johnson. Matching, Euler tours and the Chinese postman. *Mathematical programming*, 5(1):88–124, 1973.

4 Michael R. Garey and David S. Johnson. *Computers and intractability*, volume 174. Freeman San Francisco, 1979.

5 Michael Held and Richard M. Karp. The traveling-salesman problem and minimum spanning trees. *Operations Research*, 18(6):1138–1162, 1970.

6 Anna R. Karlin, Nathan Klein, and Shayan Oveis Gharan. A (slightly) improved approximation algorithm for metric TSP. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 32–45, 2021.

7 Michael Hart Moore. On the fastest route for convoy-type traffic in flowrate-constrained networks. *Transportation Science*, 10(2):113–124, 1976.

8 Dong Hwan Oh, R. Kevin Wood, and Young Hoon Lee. Optimal interdiction of a ground convoy. *Military Operations Research*, 23(2):5–18, 2018.

9 Christos H. Papadimitriou. On the complexity of edge traversing. *Journal of the ACM*, 23:544–554, 1976.

10 Marta Pascoal, M. Eugénia V. Captivo, and João C.N. Clímaco. A comprehensive survey on the quickest path problem. *Annals of Operations Research*, 147(1):5–21, 2006.

11 P.N. Ram Kumar and T.T. Narendran. A mathematical approach for variable speed convoy movement problem (CMP). *Defense & Security Analysis*, 25(2):137–155, 2009.

12 Martin Skutella. An introduction to network flows over time. In William Cook, László Lovász, and Jens Vygen, editors, *Research Trends in Combinatorial Optimization: Bonn 2008*, pages 451–482. Springer, Berlin, Heidelberg, 2009.

13 Petr Slavík. The errand scheduling problem, 1997. Technical report.

14 Ola Svensson, Jakub Tarnawski, and László A Végh. A constant-factor approximation algorithm for the asymmetric traveling salesman problem. *Journal of the ACM (JACM)*, 67(6):1–53, 2020.

15 Vera Traub and Jens Vygen. An improved approximation algorithm for the asymmetric traveling salesman problem. *SIAM Journal on Computing*, 51(1):139–173, 2022.

A Details for the proof of Lemma 11

▶ **Lemma 13.** *We have that* $\mathrm{OPT}' \in O(\mathrm{OPT})$.

Proof. Let $I' = [a, b) \subseteq I$ denote a maximally large subinterval of I such that $q_{\mathrm{OPT}}(d) = \bar{q}^{(1)}$ for each $d \in I'$. We have that $|I'| \geq L$ since $\bar{q}^{(1)}$ is the lowest speed. Due to I' we defined for each $d \in [a', \min\{a' + L, b\})$ and for each $d \in [\max\{b' - L, a\}, b')$ that $q'(d) := \bar{q}^{(1)}$. We charge the time traveled with speed $\bar{q}^{(1)}$ on $[a', \min\{a' + L, b\}) \cup [\max\{b' - L, a\}, b')$ to I'. For our charging scheme, we say that the clean interval $[a', b')$ gives $4L/\bar{q}^{(1)}$ credits to each adjacent maximally large non-clean interval, and we charge this to $[a', b')$. We do the same argumentation and charging for each interval that became clean in this iteration.

Suppose by induction that we have given as input a maximally large non-clean interval $\tilde{I}$ such that, for some k, $q_{\mathrm{OPT}}(d) \geq \bar{q}^{(k)}$ for each $d \in \tilde{I}$ that received $4L/\bar{q}^{(k-1)}$ credits from the previous iterations. Let $I' = [a, b) \subseteq \tilde{I}$ denote a maximally large subinterval of $\tilde{I}$ such that $q_{\mathrm{OPT}}(d) = \bar{q}^{(k)}$ for each $d \in I'$. Suppose that due to I' we define for each $d \in [a', \min\{a' + L, b\})$ and for each $d \in [\max\{b' - L, a\}, b')$ that $q'(d) := \bar{q}^{(k)}$.

First, if $|I'| \geq L$ then we charge the times traveled with speed $\bar{q}^{(k)}$ on $[a', \min\{a' + L, b\}) \cup [\max\{b' - L, a\}, b')$ to I'. Also, we charge I' in order to give $4L/\bar{q}^{(k)}$ credits to each adjacent maximally large non-clean interval that is adjacent to $[a', b')$.

Second, if $|I'| < L$ then I' must be at the left or the right end of $\tilde{I}$. In that case we charge those times to $2L/\bar{q}^{(k-1)}$ of the $4L/\bar{q}^{(k-1)}$ credits that we received from the previous iteration. We use the remaining $2L/\bar{q}^{(k-1)} = 4L/\bar{q}^{(k)}$ credits to give $4L/\bar{q}^{(k)}$ credits to each maximally large non-clean interval that is adjacent to $[a', b')$. Together, this implies the lemma. ◀