Hardness of Preemptive Finite Capacity Dial-a-Ride

Inge Li Gørtz*

Technical University of Denmark. Email: ilg@imm.dtu.dk.

Abstract. In the *Finite Capacity Dial-a-Ride problem* the input is a metric space, a set of objects $\{d_i\}$, each specifying a source s_i and a destination t_i, and an integer k—the capacity of the vehicle used for making the deliveries. The goal is to compute a shortest tour for the vehicle in which all objects can be delivered from their sources to their destinations while ensuring that the vehicle carries at most k objects at any point in time. In the *preemptive* version an object may be dropped at intermediate locations and picked up later
and delivered. Let N be the number of nodes in the input graph. Charikar and Raghavachari [FOCS '98] gave a $\min\{O(\log N), O(k)\}$-approximation algorithm for the preemptive version of the problem. In this paper we show that the preemptive Finite Capacity Dial-a-Ride problem has no $\min\{O(\log^{1/4-\varepsilon} N), k^{1-\varepsilon}\}$-approximation algorithm for any $\varepsilon > 0$ unless all problems in NP can be solved by randomized algorithms with expected running time $O(n^{\text{polylog}n})$.

1 Introduction

Vehicle routing and delivery problems have been widely studied in Computer Science and Operations Research. These problems occur in many practical settings such as transportation of goods or passengers and robotics (see Christofedes [5] and Golden and Assad [10]). Many of these problems are NP-hard and there has been a great deal of research in finding and analyzing heuristics to solve these problems. One such problem is the *Finite Capacity Dial-a-Ride problem*—or *Dial-a-Ride* for short—which is defined as follows. The input is a metric space, a set of objects, where each object d_i specifies a source s_i and a destination t_i, and an integer k—the capacity of the vehicle used for making the deliveries. The goal is to compute a shortest tour for the vehicle in which all objects can be delivered to their destinations (from their sources) while ensuring that the vehicle carries at most k objects at any point in time. There are two variants of the problem: the *non-preemptive* case, in which an object once loaded on the vehicle stays on it until delivered to its destination, and the *preemptive* case in which an object may be dropped at intermediate locations and then picked up later by the vehicle and delivered. The Dial-a-Ride problem generalizes the Traveling Salesman problem (TSP) even for $k = 1$ and is thus NP-hard.

* This work was performed while the author was a Ph.D. student at the IT University of Copenhagen.

Let N denote the number of nodes in the input graph, i.e., the number of points that are either sources or destinations. In this paper we show that the preemptive Dial-a-Ride problem has no $\min\{O(\log^{1/4-\varepsilon} N), k^{1-\varepsilon}\}$-approximation algorithm for any $\varepsilon > 0$ unless $\mathsf{NP} \subseteq \mathsf{ZPTIME}(n^{\text{polylog}n})$[1]. To our knowledge, the TSP lower bound—which is a small constant—was the best known so far.

The Dial-a-Ride problem has several practical applications such as transportation of elderly and/or disabled persons and courier services. In practice, multi-vehicle systems, where there are more than one vehicle, are more common. Since single-vehicle Dial-a-Ride is a special case of the multi-vehicle Dial-a-Ride problem, the hardness results in this paper holds for these problems a well.

Previous and Related Results Guan [12] proved that the preemptive case is NP-hard for trees when $k \geq 2$. Frederickson and Guan [8] showed that the unit-capacity non-preemptive case is NP-hard on trees. For this case Frederickson *et al.* [9] gave an 1.8-approximation algorithm on general graphs. The first non-trivial approximation algorithms for the Dial-a-Ride problem for general k were given by Charikar and Raghavachari [4]. For the preemptive case they gave a 2-approximation algorithm for trees. Using the results on probabilistic approximation of metric spaces by tree metrics [7] this gives an $O(\log N)$-approximation for arbitrary metrics. For the non-preemptive case they gave an $O(\sqrt{k})$-approximation algorithm for special instances on height-balanced trees. As above this implies an $O(\sqrt{k}\log N)$-approximation for arbitrary metrics. For points on a line they note that they have a 2-approximation. They also show that the ratio of the cost of the optimal non-preemptive solution to the cost of the optimal preemptive solution can be as large as $\Omega(k^{2/3})$. As noted by Charikar and Raghavachari an $O(k)$-approximation algorithm can be obtained by taking the $O(1)$-approximation algorithm for the unit-capacity case. We note that there is a simple $\frac{3N}{k}$-approximation algorithm (due to [14] for $k = N$).

Several papers have presented exact exponential time algorithms and heuristic algorithms for the Dial-a-Ride problem. For a description of many of these approaches see [6]. A related problem is the k-delivery TSP where all objects are identical and can be delivered to any of the destination points. Charikar *et al.* [3] gave a 5-approximation algorithm for both the preemptive and the non-preemptive problem. Haimovich and Rinnooy Kan [13] gave a 3-approximation for the problem when all objects initially are located at one central depot.

Our Results and Techniques Our results rely on the hardness results for the two network design problems *Buy-at-Bulk* and *SumFiber-ChooseRoute(SFCR)* (defined in the next section). Andrews [1] and Andrews and Zhang [2] showed that there is no $O(\log^{1/4-\varepsilon} N)$-approximation algorithm for uniform Buy-at-Bulk and SFCR, respectively, for any $\varepsilon > 0$ unless $\mathsf{NP} \subseteq \mathsf{ZPTIME}(n^{\text{polylog}n})$. The result for SFCR uses a network constructed from an interactive 2-prover system

[1] $\mathsf{ZPTIME}(n^{\text{polylog}n})$ is the class of problems solvable by a randomized algorithm that always returns the right answer and has expected running time $O(n^{\text{polylog}n})$, where n is the size of the input.

for MAX3SAT. They show that if the MAX3SAT formula ϕ is satisfiable then the optimal solution to the SFCR instance has small cost, and if ϕ is unsatisfiable then it has high cost. More precisely, the cost if ϕ is unsatisfiable is a factor of γ more than if ϕ is satisfiable for $\gamma = O(\log^{1/4-\varepsilon} N)$. Hence if there were an α-approximation for SFCR with $\alpha < \gamma$, then we would be able to determine if ϕ was satisfiable. Using almost the same construction we show that Buy-at-Bulk with cost function $h(x) = \lceil \frac{x}{k} \rceil$ has no $O(\log^{1/4-\varepsilon} N)$-approximation for any $\varepsilon > 0$ unless $\mathsf{NP} \subseteq \mathsf{ZPTIME}(n^{\text{polylog}n})$, when k is between $\log^{11/(8\varepsilon)-9/2} n = \Omega(\log^{1/4+(7\varepsilon)/11} N)$ and $O(2^{\log^2 n}/\log n)$. Here n is the size of ϕ. By changing some of the parameters in the construction we are able to show that the problem is not approximable within a factor of $k^{1-\varepsilon}$ for any $\varepsilon > 0$ when $k < \log^{1/4} N$.

We then show the same hardness results for the preemptive Dial-a-Ride problem by showing a relation between this problem and the Buy-at-Bulk problem with cost function $h(x)$ in the network constructed from the 2-prover system. This is the main technical contribution of this paper. Due to lack of space many proofs are omitted. They can be found in the full version of the paper [11].

2 Definitions

Uniform Buy-at-Bulk Given an undirected network $\mathcal{N}$, with lengths l_e on the edges and a set $\{(s_i, t_i)\}$ of source-destination pairs. Each pair (s_i, t_i) has an associated demand δ_i. There is a cost function f on the edges, which is a function of the amount of demand x_e using edge e. Function f is subadditive[2], and $f(0) = 0$. The goal is to route all demands δ_i from their source s_i to their destination t_i minimizing the total cost. The demands are unsplittable, i.e., demand δ_i must follow a single path from s_i to t_i. The total cost of the solution is $\sum_e f(x_e) l_e$.

SumFiber-ChooseRoute (SFCR) Here we are given $\mathcal{N}$, l_e, $\{(s_i, t_i)\}$, and δ_i as in Buy-at-Bulk. Each demand requires bandwidth equivalent to one wavelength. Each fiber can carry k wavelengths, and the cost of deploying x fibers on edge e is $x \cdot l_e$. The problem is to specify a path from s_i to t_i for all demands δ_i, and a wavelength for the demand λ_i, minimizing the total cost. Let $f_e(\lambda)$ be the number of demands assigned to wavelength λ that are routed through edge e. Then $\max_\lambda f_e(\lambda)$ is the number of fibers needed on edge e. Thus the total cost of the solution is $\sum_e l_e \max_\lambda f_e(\lambda)$.

Interactive Proof Systems A Raz-verifier is an *interactive two-prover system*. An interactive two-prover system for MAX3SAT(5) consists of a polynomial time *verifier* with access to a source of randomness and two computationally unbounded *provers*. The verifier sends a polynomial size query to each prover and receives a polynomial size answer. The provers try to convince the verifier that the formula is satisfiable. The provers cannot communicate with each other and are restricted to see only the queries addressed to them. Based on the random

[2] $f(x + y) \leq f(x) + f(y)$.

bits and the answers to the queries the verifier decides whether or not to accept the input. The verifier accepts with probability 1 if ϕ is satisfiable. If ϕ is unsatisfiable then regardless of how the provers answer the verifier accepts with a very low probability, η, called the *error probability*.

Proof System Parameters Let R be the random bits, Q_i the random query sent to prover i, and A_i the answer returned by prover i. We will use lowercase letters to denote specific values of these strings. Each random string r uniquely identifies a pair of queries q_0 and q_1. Each query may have many different answers. We say $a \in q$ if a is an answer to query q. We assume that the verifier appends the name of the prover to the query and the provers append the query name to its answer string. This way, an interaction is uniquely identified by the triple (r, a_0, a_1). If the verifier accepts the answers a_0 and a_1 from the provers we say that (r, a_0, a_1) is an *accepting interaction*. Note that two different random strings might result in the same prover-0 query (or prover-1 query), but in that case they will result in different prover-1 (prover-0) queries. Let $m(Q_i)$ denote the number of distinct possible values of Q_i. By padding random bits, we can assume, $m(Q_0) \leq m(Q_1) < 2m(Q_0)$. We can ensure that the Raz verifier has the following properties (here $|x|$ denotes the number of bits in the string x): $|R| = O(\log^2 n)$, $|Q_i| = O(\log^2 n)$, $|A_i| = O(\log^2 n)$, and $\eta = 2^{-\Omega(\log n)}$. For each i and for any $q \in \{0,1\}^{|Q_i|}$: $Pr[Q_i = q] \in \{0, 1/m(Q_i)\}$.

3 Relation between Buy-at-Bulk and Dial-a-Ride

The following lemma shows a relation between Buy-at-Bulk and Dial-a-Ride.

Lemma 1. *Let OPT_B be the value of an optimal solution to a Buy-at-Bulk instance B with source destination pairs S in graph G and cost function $h(x) = \lceil \frac{x}{k} \rceil$, and let OPT_D be the value an optimal solution to the Dial-a-Ride instance D with the same source-destination pairs S in G. Then $\mathsf{OPT}_B \leq \mathsf{OPT}_D$.*

Proof. We will abuse notation and let OPT_i stand for both the value of the optimal solution and the solution itself. We can turn OPT_D into a solution to instance B as follows: Route a demand δ_i from its source s_i to its destination t_i by the same edges as object δ_i passes in OPT_D. valid solution. It is straightforward to verify that the cost of this solution is no larger than OPT_D. □

Since the optimal solution to B might be disconnected, there is in general no way to turn OPT_B into a solution to D at a cost bounded in terms of OPT_B. However, on the network used to construct the hardness result for Buy-at-Bulk we will show that in the case were the Max3Sat instance ϕ is satisfiable it is possible to turn the solution to B into a solution to D at cost at most $7 \cdot \mathsf{OPT}_B$.

4 The Network

In this section we describe the network that is used to show hardness of SFCR in [2]. The network is constructed randomly from an interactive proof system

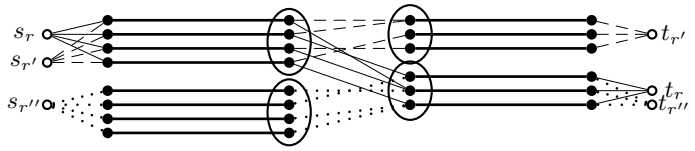

Fig. 1. The basic network $\mathcal{N}_0$. For each of the three random strings r, r' and r'', four canonical paths corresponding to four accepting interactions, are shown (r solid, r' dashed, and r'' dotted). The long thick edges are the answer edges.

for Max3Sat. The idea is for each demand to define a set of *canonical paths* on which the demand can be carried. These canonical paths correspond to accepting interactions and are short paths directly connecting the source and destination.

We first construct a basic network $\mathcal{N}_0$, which is used as the base case in the random construction. Given an instance ϕ, first construct the interactive two-prover system. This is then turned into an instance of SFCR as follows. For each possible answer a there is an *answer edge* (also denoted by a). For each random string r there is a source node s_r, a destination node t_r, and a demand d_r of one to be routed from s_r to t_r. For each accepting interaction (r, a_0, a_1) there is a canonical path p. This path starts at node s_r, passes through a_0 and a_1 and ends at t_r. To make this possible we place edges between s_r and a_0, between a_0 and a_1, and between a_1 and t_r. The edge between a_0 and a_1 is referred to as a *center edge*, and the edge between s_r and a_0, and between a_1 and t_r as a *demand edge*. For each query q the answer edges $a \in q$ are grouped together (see Fig. 1). Answer edges have length $h > 1$ and the other edges have length 1.

Before defining the final network, we define a random network $\mathcal{N}_1$ in terms of $\mathcal{N}_0$ and two parameters X and Z. The network essentially replicates $\mathcal{N}_0$ in the vertical direction XZ times. Each answer edge a_0 (resp. a_1) of $\mathcal{N}_0$ has XZ copies, denoted by $a_{0,x,z}$ ($a_{1,x,z}$) where $0 \le x < X$ and $0 \le z < Z$. For each random string r, create X demands $d_{r,x}$ and X source and destination nodes, $s_{r,x}$ and $t_{r,x}$, where $0 \le x < X$. Each of the X demands $d_{r,x}$ routes one unit of flow from $s_{r,x}$ to $t_{r,x}$. For each accepting interaction (r, a_0, a_1), the demand $d_{r,x}$ has a canonical path that starts at $s_{r,x}$, passes through $a_{0,x',z'}$ and $a_{1,x'',z''}$ and ends at $t_{r,x}$. The answer edges $a_{0,x',z'}$ and $a_{1,x'',z''}$ are chosen randomly. More precisely, x' and x'' are chosen uniformly at random from the range $\{0, 1, \ldots, X-1\}$ and z' and z'' are chosen uniformly at random from the range $\{0, 1, \ldots, Z-1\}$. To make the canonical paths feasible, $\mathcal{N}_1$ has center edges connecting $a_{0,x',z'}$ and $a_{1,x'',z''}$, and edges connecting $s_{r,x}$ to $a_{0,x',z'}$, and $a_{1,x'',z''}$ to $t_{r,x}$.

The final network $\mathcal{N}_2$ is essentially a concatenation of $\mathcal{N}_1$ in the horizontal direction Y times for some parameter Y, where each level is constructed randomly and independently. Each answer edge is indexed by $a_{0,x,z,y}$ (resp. $a_{1,x,z,y}$) where $y \in \{0, 1, \ldots, Y-1\}$. As in $\mathcal{N}_1$, X demands $d_{r,x}$, $0 \le x < X$, are created for each random string r. For each accepting interaction (r, a_0, a_1), the demand $d_{r,x}$ has a canonical path starting at $s_{r,x}$ followed by answer edges $a_{0,x,z,0}$ and $a_{1,x,z,0}$ chosen uniformly at random at level $y = 0$. At each subsequent level y, the answer edges are chosen uniformly at random until the path ends at $t_{r,x}$.

The center edges and demand edges are defined by the canonical paths. Each canonical path also requires an edge between each consecutive pair of levels.

5 Hardness of Buy-at-Bulk with Cost Function $\lceil \frac{x}{k} \rceil$

In this section we use the network $\mathcal{N}_2$ to show hardness of Buy-at-Bulk with cost function $\lceil \frac{x}{k} \rceil$. The results are obtained by changing some of the parameters in the network compared to paper by Andrews and Zhang [2], but otherwise the proofs in this section are similar to the ones in the [2]. We use the following parameters to show hardness with dependence on N.

- $\ell = \log^{\alpha} n$ for some constant α.
- $Z = \frac{2^{|r|}}{k \min\{m(Q_0), m(Q_1)\}}$
- $X = (2^{6+|r|+|a_0|+|a_1|} YZ)^{2l+1} = 2^{O(\log^{\alpha+2} n)}$
- $k = \log^{\frac{\alpha}{4}+4} n$
- $\sigma = \log^{\frac{\alpha}{4}} n$
- $Y = \sqrt{\ell} = \log^{\frac{\alpha}{2}} n$
- $h = \frac{2^{|r|}}{(m(Q_0)+m(Q_1))Z}$
- $\eta = \frac{1}{\sigma^2 \log n}$

The only parameter changed compared to [2] is h. To show hardness with dependence on k we allow k to be smaller than $\log^{\alpha/4+4} n$. To make the proofs go through we change Z and h as follows. Let $c > 1$ be a constant such that $k = \log^{\alpha/4+4} n/c$ and set

- $Z = \frac{2^{|r|}}{ck \min\{m(Q_0), m(Q_1)\}} = \frac{2^{|r|}}{\log^{\frac{\alpha}{4}+4} n \cdot \min\{m(Q_0), m(Q_1)\}}$
- $h = \frac{2^{|r|}}{c(m(Q_0)+m(Q_1))Z}$

The next two lemmas hold for both definitions of Z and h. An answer edge is said to be *bought* if any demand is routed through it.

Lemma 2. *If ϕ is satisfiable, then the Buy-at-Bulk instance has a solution of total cost at most $2^{|r|}(2Y+1)X + 2(m(Q_0)+m(Q_1))hXYZ$.*

Proof. Since ϕ is satisfiable there are two provers that always cause the verifier to accept. We route the demand on answer edge a if and only if for these two provers a is the answer to query q. For each string r there must be some accepting interaction (r, a_0, a_1) for which both a_0 and a_1 have been bought. Each of the demands $d_{r,x}$, for $0 \leq x < X$, has one canonical path that corresponds to (r, a_0, a_1). The demand $d_{r,x}$ is routed along this path. There are $2Y+1$ length one edges on this path and thus the total number of edges of length one needed is at most $2^{|r|}(2Y+1)X$. It is possible to show that the expected cost of an answer edges is two. The details are omitted due to lack of space. The expected total cost of the answer edges is therefore $2XZY(m(Q_0)+m(Q_1))h$. The total solution has expected cost $2^{|r|}(2Y+1)X + 2(m(Q_0)+m(Q_1))hXYZ$, and the cost of the optimal solution must therefore have cost no higher than that. □

The second lemma gives a lower bound on the cost of the solution when ϕ is unsatisfiable. The proof is omitted due to lack of space.

Lemma 3. *With probability $\frac{2}{3} - o(1)$, if the instance ϕ of 3SAT is unsatisfiable then the cost of any solution to our instance of Buy-at-Bulk is at least*

$$\min\{\frac{\sigma h}{10}(m(Q_0)+m(Q_1))XYZ, \frac{Y^2}{4k}\big((X2^{|r|})(1-\frac{77}{375}-o(1))-X\big)\}.$$

Combining Lemma 2 and 3 we get the following hardness result for Buy-at-Bulk with cost function $h(x) = \lceil \frac{x}{k} \rceil$. The proof is omitted due to lack of space.

Corollary 1. *For any $\varepsilon > 0$, there is no $\min\{O(\log^{\frac{1}{4}-\varepsilon} N), k^{1-\varepsilon}\}$-approximation algorithm for Buy-at-Bulk with cost function $h(x) = \lceil \frac{x}{k} \rceil$ unless all problems in NP can be solved by a randomized algorithm with expected running time $O(n^{\text{polylog } n})$.*

6 Routing in the Network

Let B be the instance of Buy-at-Bulk constructed in Section 5, and let D be an instance of preemptive Dial-a-Ride with the same source-destination pairs in the same network. Let SOL_B denote the solution used to give the bound on the cost of the optimal solution in Lemma 2, and let OPT_D be the optimal solution to D. In this section we show how to construct a solution to D of cost at most $7 \cdot \mathsf{SOL}_B$ when ϕ is satisfiable.

Let $\mathcal{N}_2^f$ be the network induced by the edges bought in SOL_B. Recall that in SOL_B all demands are routed on canonical paths. For each demand d, let p_d be the canonical path which d is routed on in SOL_B. We say that edge $e \in \mathcal{N}_2^f$ is *used* by an object d if e is on the path p_d. Let u_e be all the objects using edge e.

6.1 The Tour when $\mathcal{N}_2^f$ is Connected

We will first explain how to construct the tour when $\mathcal{N}_2^f$ is connected. We will say that the tour is using an edge in the forward direction if it uses it in the same direction as the demands routed on this edge and backwards otherwise. Assume that any edge in $\mathcal{N}_2^f$ is used by at most k objects (we show later how to get rid of this assumption). We will ensure that the tour has the following properties:

(i) The tour only uses edges from $\mathcal{N}_2^f$.
(ii) An object d will only be in the vehicle when the vehicle is on an edge $e \in p_d$.
(iii) When the vehicle goes forward on an edge it is either empty or carries all objects using that edge.

The algorithm to construct the tour has two kinds of phases—a *delivery phase* and a *pickup phase*—which are intermixed. In a delivery phase we are in the process of delivering a certain object. In a pickup phase the vehicle is on its way to pick up the next object to be delivered. The vehicle is always empty in a pickup phase. The algorithm calls the following two procedures.

Deliver(d,s): Follow p_d. For each edge on p_d there are two cases:

1. All objects from u_e are present at u: Pick up all the objects and traverse e. At node v drop off all objects not going in the same direction as d.
2. One or more objects from u_e are not present at u: Drop off d at node u, and go to pick up these objects as follows. Let d' be such an object. Follow $p_{d'}$ backwards from e until encountering d'. Pick up d' and deliver d' at *node* u (not $s_{d'}$) by recursively calling Deliver(d',u).

Route(d): First deliver d by calling Deliver(d,s_d) (this is the *delivery phase for object d*). Then follow the route constructed during this call to Deliver backwards until d_d is reached (this is a *pickup phase*). Whenever encountering an undelivered object d' on the way, pick it up and deliver it to its destination by recursively calling Route(d').

Algorithm The algorithm starts at a node $s_{r,x}$ for some r and x, pick up $d_{r,x}$ and call Route($d_{r,x}$). Below we will show that when the vehicle returns to $d_{r,x}$ all objects are delivered.

Analysis of the algorithm It is easy to verify that the tour made by the algorithm satisfies property (i), (ii), and (iii). We will denote the route constructed during the delivery phase for object d by r_d.

Lemma 4. *For any object d, the route r_d, has the following properties:*

(iv) *r_d only goes backwards on an edge e to fetch "missing" objects. If d' is such an object then $e \in p_{d'}$.*
(v) *If r_d goes backwards on edge e it returns to the right endpoint of e through e.*
(vi) *When route r_d traverses an edge e in the forward direction the vehicle contains all objects using e.*

Proof. Property (iv) and (vi) follows immediately from the description of the algorithm. It remains to prove property (v). All canonical paths go through all levels of the network in increasing order. Therefore an object missing at the left endpoint of some edge at level i can be fetched at a level smaller than i or at i if the edge is not the first edge on level i. It is thus possible to fetch all objects missing at a certain node, since there are no cyclic dependencies. □

Lemma 4 gives us the following two corollaries.

Corollary 2. *For any object d, the route r_d traverses each edge in $\mathcal{N}_2^f$ at most once in each direction.*

Corollary 3. *For any two objects d_1 and d_2 the routes r_{d_1} and r_{d_2} are disjoint.*

Lemma 5. *All objects are delivered to their destination.*

Proof. By contradiction. Recall, we assumed $\mathcal{N}_2^f$ is connected. Assume some subset of objects S are not delivered. Consider an object $d \in S$. If d is at a node $u \neq s_d$ then it was left at u during the delivery phase of some object d'. But then it would have been picked up and delivered to its destination when the vehicle traversed $r_{d'}$ backwards. Thus d must still be at its source s_d. Since d is still at s_d the path p_d does not share any edges with any path $p_{d'}$ where d' is a delivered object. To see this assume d shared an edge e with a delivered object d'. Due to property (ii) the vehicle crossed e containing d', since d' is delivered. Due to property (vi) of Lemma 4 d must have been in the vehicle when it crossed e, and thus d would no longer be at s_d. Since SOL_B are using canonical paths for

each object, the graph $\mathcal{N}_2^f$ has the property that if two canonical paths p_d and $p_{d''}$ meet at some vertex then they must share an edge adjacent to that vertex. Therefore p_d cannot share any vertices with any path $p_{d'}$ where d' is a delivered object. This is true for all objects $d \in S$, contradicting that $\mathcal{N}_2^f$ is connected. □

Lemma 6. *When $\mathcal{N}_2^f$ is connected the tour has length at most $4 \cdot \mathsf{SOL}_B$.*

Proof. Let $l(r_d)$ denote the length of the route r_d. The total length of the parts of the tour constructed during delivery phases is $\sum_{d \in D} l(r_d)$.

Now consider the parts of the tour constructed during a pickup phase. Here we are going backwards on the route r_d for some object d. During this pickup phase we stop each time we meet an object d' and deliver it by calling Route(d'). Due to Corollary 3 the part of the tour constructed during the call to Route(d') is disjoint from r_d, since it only contains edges on $r_{d'}$. The route r_d is thus traversed at most once during the pickup phases. Thus the total length of the parts of the tour constructed during delivery phases is at most $\sum_{d \in D} l(r_d)$.

Adding together the total length of the tours constructed during the delivery phases and the pickup phases, we get that the total length of the tour is at most $2 \cdot \sum_{d \in D} l(r_d)$. Using Corollary 2 and Corollary 3 we get that the tour uses each edge in $\mathcal{N}_2^f$ at most 4 times, and thus the cost of the tour is at most $4 \cdot \mathsf{SOL}_B$. □

Edges used by more than k objects We assumed that any edge in $\mathcal{N}_2^f$ is used by at most k objects. We can get rid of this assumption by a minor modification of the algorithm. Let S_e be the set of objects using edge e. Then the solution SOL_B paid $\lceil \frac{S_e}{k} \rceil \cdot l_e$ for this edge. As before, when we want to traverse e we go backwards and pick up all objects in S_e. We then go forward and back on e carrying as many objects from S_e as possible each time until all objects from S_e are on the right endpoint of e. The number of times we traverse e is $\lceil \frac{S_e}{k} \rceil$, and thus Lemma 6 still holds.

6.2 $\mathcal{N}_2^c$ Connected and $\mathcal{N}_2^f$ Disconnected

Let $\mathcal{N}_2^c$ be the graph induced by the canonical paths ($\mathcal{N}_2$ can contain answer edges that are not part of any canonical path). If $\mathcal{N}_2^c$ is connected but $\mathcal{N}_2^f$ is disconnected we can add edges from $\mathcal{N}_2^c$ to $\mathcal{N}_2^f$ to connect it. We can do this by adding edges of total length equal to the number of connected components minus one times the length of a canonical path in $\mathcal{N}_2^c$.

First we note that since $\mathcal{N}_2^c$ consists of the union of canonical paths, then for any component C in $\mathcal{N}_2^f$ there must be another component C' in $\mathcal{N}_2^f$ such that some object d routed in C has a canonical path p that intersect with a canonical path p' for an object d' routed in C'. We connect C and C' by adding the following edges: All edges on p from s_d to the intersecting edge e (including e), and all edges on p' from e to $t_{d'}$. We call these added edges a *connecting path from C' to C*. Since $\mathcal{N}_2^c$ is connected we can make $\mathcal{N}_2^f$ connected by adding $c-1$ connecting paths, where c is the number of connected components in $\mathcal{N}_2^f$. We

add these connecting paths in such a way that all components can be reached from one component—called the *start component*—using a path that when going from component C to a component C' uses a connecting path from C to C' (not from C' to C). Since the length of a connecting path is the same as the length of a canonical path the total length is $c-1$ times the length of a canonical path. Since each connected component consists of at least one canonical path the total length of the connecting paths is at most the same as the sum of all edges in $\mathcal{N}_2^f$, i.e., SOL_B.

Constructing the tour Start in the start component C_s in $\mathcal{N}_2^f$ and deliver the objects in this component as described in the previous section. Whenever the vehicle gets to a node d_d which is the starting point of a connecting path from this component to another component C, it follows this connecting path to C and delivers the objects in C the same way. When all objects in a component are delivered the vehicle returns to the starting point in this component and from there to the previous component C' if such a component exists. It then carries on delivering the objects in C'.

Lemma 7. *When $\mathcal{N}_2^c$ is connected the tour has length at most $6 \cdot \mathsf{SOL}_B$.*

Proof. If $\mathcal{N}_2^f$ is connected it follows from Lemma 6. If $\mathcal{N}_2^f$ is disconnected we use the approach described above. To deliver the objects in a single component we use no more time than in the previous section. By Lemma 6 the contribution from these parts of the tour is at most $4 \cdot \mathsf{SOL}_B$ in total. To get to the next component and back again we use a connecting path and the sum of the edges used to get to and from connected components is thus at most $2 \cdot \mathsf{SOL}_B$. □

6.3 $\mathcal{N}_2^c$ Disconnected

If $\mathcal{N}_2^c$ is disconnected we connect it by adding edges of length one between a source node in one component and a source node in another component . We call these edges *component* edges. We add the minimum number of component edges, i.e., $l-1$ where l is the number of connected components. This can be seen as constructing a tree on the components.

Since we add the component edges between disjoint components in $\mathcal{N}_2^c$, which are also disjoint components in $\mathcal{N}_2$, we do not introduce any new cycles in $\mathcal{N}_2$. Therefore the component edges cannot decrease the cost of the optimal solution to the Buy-at-Bulk instance or to the Dial-a-Ride instance: Let C_1 and C_2 be two components connected by a component edge e. If some object d with source s_d in C_1 is using e, then it has to use it again to get back to C_1, since $s_d \in C_1$ and the only connection between C_1 and C_2 is e.

Constructing the Tour The vehicle first delivers the objects in a component C in $\mathcal{N}_2^c$ as described in the previous section. When it gets to the source node in the component that has a component edge to a source node in another component C', it goes to C' and delivers the objects in C' the same way. When all objects

in a component are delivered it returns to the starting point of this component and follows the component edge back to the previous component C if such a component exists. It then carries on delivering the objects in component C.

Lemma 8. *The optimal solution to D has cost at most $7 \cdot \mathsf{OPT}_B$.*

Proof. The cost of delivering the objects in the original components of $\mathcal{N}_2$ is at most $6 \cdot \mathsf{SOL}_B$ due to Lemma 7. The total length of the new edges is $l-1$ which is less than $1/2 \cdot \mathsf{SOL}_B$, since each connected component has a canonical path of at least three. The new edges are used twice: once in each direction. □

7 Hardness of Preemptive Dial-a-Ride

From Lemma 8 and Lemma 2 we get,

Lemma 9. *If ϕ is satisfiable, then the Dial-a-Ride instance has a solution of total cost $7 \cdot 2^{|r|}(2Y+1)X + 2(m(Q_0)+m(Q_1))hXYZ$.*

We can now use Lemma 1, Lemma 3, and Lemma 9 to show hardness of the Dial-a-Ride problem.

Lemma 10. *Let $\gamma = \log^{\frac{\alpha}{4}-5} n$. If there exists a γ-approximation algorithm for the Finite Capacity Dial-a-Ride problem, then there exists a randomized $O(n^{\text{polylog } n})$ time algorithm for 3SAT.*

Proof. For any 3SAT instance ϕ we construct the network $\mathcal{N}_2$ from the two-prover system and then apply a γ-approximation algorithm A for Dial-a-Ride.

If the 3SAT instance ϕ is satisfiable then by Lemma 9 and our choice of h there is a solution to our instance of Dial-a-Ride of cost at most $7 \cdot 2^{|r|}(2Y+1)X + 2(m(Q_0)+m(Q_1))hXYZ = 7 \cdot 2^{|r|}(4Y+1)X$. Hence, the γ-approximation algorithm returns a solution of cost at most $\gamma \cdot 7 \cdot 2^{|r|}(4Y+1)X$, and we declare ϕ satisfiable. If ϕ is unsatisfiable then by Lemma 1, Lemma 3 and our choice of h, with probability $2/3 - o(1)$, any solution have cost at least the minimum of $\Omega(\sigma 2^{|r|}XY)$ and $\Omega(\frac{\ell}{k}X2^{|r|})$. Both these expressions are strictly larger than $\gamma \cdot 7 \cdot 2^{|r|}(4Y+1)X$.

The construction of the network takes time $O(n^{\text{polylog } n})$ since $\mathcal{N}_2$ has size $O(n^{\text{polylog } n})$. Hence we have described a randomized $O(n^{\text{polylog } n})$ time algorithm for 3SAT that has one-sided error probability at most $1/3 + o(1)$. It is possible to convert this into a randomized algorithm that never makes an error and has expected running time $O(n^{\text{polylog } n})$. □

In the Dial-a-Ride instance N is the number of sources and destinations. We have $2^{|r|}X$ sources and $2^{|r|}X$ destinations, and thus $N = 2 \cdot 2^{|r|}X = 2^{O(\log^{\alpha+2} n)}$. For any constant $\varepsilon > 0$, if we set $\alpha = \frac{11}{2\varepsilon} - 2$ then $\gamma = \Omega(\log^{1/4-\varepsilon} N)$. This gives us the following corollary.

Corollary 4. *There is no $O(\log^{\frac{1}{4}-\varepsilon} N)$-approximation algorithm to the preemptive Finite Capacity Dial-a-Ride problem on general graphs for any constant $\varepsilon > 0$ unless $\mathsf{NP} \subseteq \mathsf{ZPTIME}(n^{\text{polylog}n})$.*

In the above construction we had $k = \log^{\alpha/4+4} n$. The proofs hold for larger k too, but since Z should be a positive integer we require $k \leq 2^{|r|}/\min(m(Q_0), m(Q_1))$. To get a hardness result for small k we chang the variables Z and h as described in Section 5. Using Lemma 1 and Lemma 3, we get

Lemma 11. *Let $k < \log^{\frac{1}{4}} N$. Then there is no $k^{1-\varepsilon}$-approximation algorithm to the preemptive Finite Capacity Dial-a-Ride problem on general graphs for any constant $\varepsilon > 0$ unless $\mathsf{NP} \subseteq \mathsf{ZPTIME}(n^{\text{polylog}n})$.*

The proof is omitted due to lack of space. To summarize we have shown,

Theorem 1. *There is no $\min\{O(\log^{\frac{1}{4}-\varepsilon} N), k^{1-\varepsilon}\}$-approximation algorithm to the preemptive Finite Capacity Dial-a-Ride problem on general graphs for any constant $\varepsilon > 0$ unless $\mathsf{NP} \subseteq \mathsf{ZPTIME}(n^{\text{polylog}n})$.*

Acknowledgments The author wants to thank Moses Charikar and Matthew Andrews for many helpful and useful discussions.

References

1. M. Andrews. Hardness of buy-at-bulk network design. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 115–124, October 2004.
2. M. Andrews and L. Zhang. Bounds on fiber minimization in optical networks with fixed fiber capacity. In *IEEE INFOCOM*, 2005.
3. M. Charikar, S. Khuller, and B. Raghavachari. Algorithms for capacitated vehicle routing. *SICOMP: SIAM Journal on Computing*, 31(3):665–682, 2002.
4. M. Charikar and B. Raghavachari. The finite capacity dial-a-ride problem. In *IEEE Symposium on Foundations of Computer Science*, pages 458–467, 1998.
5. N. Christofedes. Vehicle routing. In E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, editors, *The Traveling Salesman Problem*, pages 431–448. John Wiley & Sons, 1985.
6. G. Desaulniers, J. Desrosiers, A. Erdmann, M. M. Solomon, and F. Soumis. VRP with pickup and delivery. In P. Toth and D. Vigo, editors, *The vehicle routing problem*, pages 225–242. Society for Industrial and Applied Mathematics, 2001.
7. J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. System Sci.*, 69(3):385–497, 2004.
8. G. N. Frederickson and D. J. Guan. Non-preemptive ensemble motion planning on a tree. *Journal of Algorithms*, 15(1):29–60, 1993.
9. G. N. Frederickson, M. S. Hecht, and C. E. Kim. Approximation algorithms for some routing problems. *SIAM Journal on Computing*, 7(2):178–193, 1978, May.
10. B. L. Golden and A. A. Assad. *Vehicle Routing: Methods and Studies*. Studies in Management Science and Systems, 16. Elsevier, 1991.
11. I. L. Gørtz. Hardness of preemptive finite capacity dial-a-ride. IMADA Preprints 2006 No. 4, University of Southern Denmark, 2006.
12. D. J. Guan. Routing a vehicle of capacity greater than one. *Discrete Applied Mathematics*, 81(1-3), 1998.
13. M. Haimovich and A. H. G. Rinnooy Kan. Bounds and heuristics for capacitated routing problems. *Mathematics of Operations Research*, 10(4):527–542, 1985.
14. H. N. Psaraftis. An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows. *Transportation Science*, 17(3):351–357, 1983.