Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

DIETRICH HAUPTMEIER SVEN O. KRUMKE JÖRG RAMBAU
HANS-CHRISTOPH WIRTH

Euler is Standing in Line

Dial-a-Ride Problems with FIFO-Precedence-Constraints

Preprint SC 99-06 (March 1999)

EULER IS STANDING IN LINE
DIAL-A-RIDE PROBLEMS WITH FIFO-PRECEDENCE-CONSTRAINTS

D. HAUPTMEIER, S. O. KRUMKE, J. RAMBAU, AND H.-C. WIRTH

ABSTRACT. In this paper we study algorithms for "Dial-a-Ride" transportation problems. In the basic version of the problem we are given transportation jobs between the vertices of a graph and the goal is to find a shortest transportation that serves all the jobs. This problem is known to be NP-hard even on trees. We consider the extension when precedence relations between the jobs with the same source are given. Our results include a polynomial time algorithm on paths and an approximation algorithm on general graphs with a performance of 9/4. For trees we improve the performance to 5/3.

1. INTRODUCTION AND OVERVIEW

Transportation problems where objects are to be transported between given sources and destinations in a metric space are classical problems in combinatorial optimization. Applications include the routing of pick-up-and-delivery vehicles, the control of automatic storage systems and scheduling of elevators. This leads to the following optimization problem (DARP): We are given transportation jobs between the vertices of a graph and the goal is to find a shortest transportation that serves all the jobs.

A natural extension of DARP is the addition of precedence constraints between the jobs that start at the same vertex. This variant is motivated by applications in which first-in-first-out (FIFO) waiting lines are present at the sources of the transportation jobs. In this case, jobs can be served only according to their order in the line. Examples with FIFO-lines are cargo elevator systems where at each floor conveyor belts deliver the goods to be transported. Elevators also motivate the restriction of DARP to paths, i.e., to the case where the underlying graph forms a path.

We show that in the case of FIFO-precedence relations, the problem can be solved in polynomial time on paths. On trees, however, the problem becomes NP-hard. We present an approximation algorithm which works on general graphs and which has performance $(\rho_{\text{TSP}} + 3)/2$, where ρ_{TSP} is the performance of the best approximation for the TSP with triangle inequality. Using Christofides' algorithm for the TSP this yields a performance of $9/4$. For trees we improve this performance to $5/3$ by modifiying our algorithm appropriately.

We also show how to extend our results to the case when there are start- and stop-penalties, which makes the problem more realistic in view of applications.

Key words and phrases. Vehicle Routing, Elevator system, Eulerian Cycle, Approximation Algorithms.

Research supported by the German Science Foundation (DFG, grant Gr 883/5-1).

In elevator systems the time that the elevator needs to accelerate or decelerate in order to pick up or deliver persons (or cargo) can usually not be neglected. Thus, it is natural to penalize each stop and start of the server on its route.

Start- and stop-penalties do not introduce a completely new situation: the problem with penalties can be modelled as DARP on a slightly larger graph. However, the penalties change the complexity of the problem when restricted to the simplest class of graphs. We prove that the problem with penalties becomes NP-hard even on paths without any precedence constraints, which contrasts with the polynomial solvability of the problem without penalties.

This paper is organized as follows. In Section 2 we formally state the problem DARP and introduce notation. We also show that DARP can be equivalently formulated as a graph augmentation problem. This key observation will be used to design our algorithms. In Section 5 we prove structural facts about Eulerian cycles in a graph that respect a given "FIFO-order" on the arcs. In Section 6 we prove hardness results. Section 7 contains a polynomial algorithm for paths. In Section 8 we present approximation algorithms for general graphs and trees. Section 9 discusses the extension to start- and stop-penalties.

2. Preliminaries and Problem Formulations

A *mixed graph* $G = (V, E, A)$ consists of a set V of vertices, a set E of undirected edges, and a set A of directed arcs (parallel arcs allowed). An edge with endpoints u and v will be denoted by $[u, v]$, an arc from u to v by (u, v). We denote by $n := |V|$, $m_E := |E|$ and $m_A := |A|$ the number of vertices, edges and arcs, respectively. For a vertex $v \in V$ we let A_v be the set of arcs in A emanating from v.

If $X \subseteq E \cup A$, then we denote by $G[X]$ the *subgraph of G induced by X*, that is, the subgraph of G consisting of the arcs and edges in X together with their endpoints. Throughout the paper we assume that $G[E]$ is connected and contains all endpoints of arcs from A. The *out-degree* of a vertex v in G, denoted by $d^+_G(v)$, equals the number of arcs in G leaving v. Similarly, the *in-degree* $d^-_G(v)$ is defined to be the number of arcs entering v.

If $X \subseteq A$ we briefly write $d^+_X(v)$ and $d^-_X(v)$ instead of $d^+_{G[X]}(v)$ and $d^-_{G[X]}(v)$. A graph G is called *degree balanced* if $d^+_G(v) = d^-_G(v)$ for all vertices $v \in V$. A *closed walk* in the mixed graph $G = (V, E, A)$ is a cycle which may visit vertices, edges and arcs multiple times.

A *directed spanning tree rooted towards $o \in V$* is a subgraph $D = (V, Y)$ of a directed graph $H = (V, R)$ which is a tree and which has the property that for each $v \in V$ it contains a directed path from v to o.

Since most of the problems under study are NP-hard, we are interested in approximation algorithms for them. Let Π be a minimization problem. A polynomial-time algorithm A is said to be a *ρ-approximation algorithm* for Π, if for every problem instance I of Π with optimal solution value $\mathrm{OPT}(I)$ the solution of value $\mathsf{A}(I)$ returned by the algorithm satisfies $\mathsf{A}(I) \leq \rho \cdot \mathrm{OPT}(I)$.

2.1. **Basic Problem.** In the "Dial-a-Ride Problem" DARP we are given a finite set of locations and a finite set of transportation jobs, where each job is a pair of locations, the source and the target of the job. The problem consists of finding a shortest transportation for the jobs starting and ending at a designated start location. More formally, we can define DARP as follows:

Definition 2.1 (Dial-a-Ride Problem (DARP))**.** The input for DARP consists of a finite mixed graph $G = (V, E, A)$, an origin vertex $o \in V$ and a nonnegative weight function $c\colon E \cup A \to \mathbb{R}_{\geq 0}$. It is assumed that for any arc $a = (u, v) \in A$ its cost $c(a)$ equals the length of a shortest path from u to v in $G[E]$.

The goal of DARP is to find a closed walk in G of minimum cost which starts in o and traverses each arc in A.

An important observation is that DARP can be equivalently formulated as a *graph augmentation problem*. Let $A(E)$ be the set of arcs such that for each undirected edge $e \in E$ the set $A(E)$ contains an anti-parallel pair of arcs between the endpoints of e. We can then extend the cost function c to $A(E)$ in a natural way by defining the cost of an arc in $A(E)$ to be the cost of the corresponding undirected edge in E.

Let W be any feasible solution to a given instance of DARP, i.e., a closed walk that starts in o and traverses each arc in A. Then W induces a multiset S of arcs in the following way: For each time an undirected edge $e = [u, v] \in E$ is traversed by W from u to v, the set S contains a copy of the directed arc (u, v). The graph $G[A \cup S]$ consisting of the arcs in $A \cup S$ and their endpoints (which include the origin o) is then Eulerian. This follows, since tracing W and for each undirected edge in W traversing the corresponding directed arc in S in the respective direction yields an Eulerian cycle.

Conversely, let S be a multiset of arcs from $A(E)$ such that $G[A \cup S]$ is Eulerian and includes the origin o. Then we can easily obtain a feasible solution W for our instance of DARP as follows: Choose an Eulerian cycle C in $G[A \cup S]$ which starts and ends in o. The walk W starts at o and then follows C. If the current arc r from C is in A then W traverses this arc in G, otherwise W traverses the undirected edge corresponding to r.

Thus, any feasible solution for DARP corresponds to an augmenting set S of arcs such that $G[A \cup S]$ is Eulerian and contains o and vice versa. This enables us to reformulate DARP equivalently as the problem of finding a multiset $S \subseteq A(E)$ minimizing the weight $c(A \cup S)$ such that $G[A \cup S]$ is Eulerian and includes o.

2.2. **Precedence Constraints.** FIFO-DARP is an extension of DARP. For each vertex $v \in V$ we are additionally given a partial order $\prec_v$ on the arcs in A_v. For each feasible solution we require that the arcs from A_v are traversed according to that partial order: whenever $a \prec_v a'$, then a must be traversed before a' in any feasible solution.

A partial order $\prec$ on the arc set of a graph $H = (V, R)$ is a *FIFO-order*, if it satisfies: $r \prec r'$ implies that r and r' have the same source. The partial order $\prec$ on the arc set A of $G = (V, E, A)$ resulting from the disjoint union of the partial orders $\prec_v$ is clearly a FIFO-order. In the sequel $\prec$ is extended to $A \cup A(E)$ by defining that arcs from $A(E)$ are incomparable to each other and to those of A.

FIFO-orders are useful to model situations in applications, when FIFO waiting lines are present at each source and objects can be picked only from the head of the queue. Figure 1 shows an example where a FIFO-respecting transportation is strictly longer than a transportation neglecting the FIFO-order. The undirected edges of the graph (which form a path) are drawn as solid lines, and the arcs corresponding to the transportation jobs are shown as dashed arcs. If no constraints have to be obeyed, then the jobs can be served without any "empty move", i.e., without traversing any undirected edge. If the constraint $a' \prec a$ must be obeyed then two empty moves are necessary.

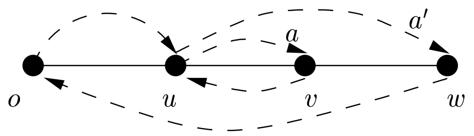

FIGURE 1. One precedence constraint increases the cost.

Again, FIFO-DARP can be reformulated as a graph augmentation problem. To do this, we need some additional notations:

Definition 2.2 ($\prec$-respecting Eulerian Cycle, $\prec$-Eulerian)**.** Let $H = (V, R)$ be a directed graph, $\prec$ be a FIFO-order on the arcs R, and $o \in V$. A *$\prec$-respecting Eulerian cycle* in H *with start* o is a Eulerian Cycle C in G such that $a \prec a'$ implies that in the walk from o along C the arc a appears before a'. The graph H is then called *$\prec$-Eulerian with start* o.

Notice that in contrast to the case of classical Eulerian cycles, for $\prec$-respecting Eulerian cycles it is meaningful to specify a start node explicitly. Consider the graph in Figure 1, with solid edges removed. Then, for $a \prec a'$, there is a $\prec$-respecting Eulerian cycle with start vertex o, but there is none starting at v.

Definition 2.3 (Graph Augmentation Version of FIFO-DARP)**.** An instance of the problem FIFO-DARP consists of the same input as for DARP and additionally a FIFO-order $\prec$ on the arc set A. The goal is to find a multiset S of arcs from $A(E)$ minimizing the weight $c(A \cup S)$ such that $G[A \cup S\,]$ is $\prec$-Eulerian with start o and to determine a $\prec$-respecting Eulerian cycle in $G[A \cup S]$.

In the sequel we consider FIFO-DARP as a graph augmentation problem. We use S^* to denote an optimal solution and $\text{OPT} := c(A \cup S^*)$ to denote its cost. Notice that if C^* is a $\prec$-respecting Eulerian cycle in $G[A \cup S^*]$ with start o, then the length of C^* is equal to OPT.

2.3. **Related Work.** The problem DARP is also called the Stacker-Crane-Problem. In [FG93] it is shown that the problem is NP-hard even on trees, i.e., if the graph $G[E]$ induced by the edges in the mixed graph $G = (V, E, A)$ is a tree. In [FHK78] the authors present a 9/5-approximation algorithm for the problem on general graphs. An improved algorithm for trees with performance 5/4 is given in [FG93]. On paths DARP can be solved in polynomial time [AK88].

Precedence constraints have been studied in the case of Chinese Postman tours in [DST87] (Recall that the Chinese Postman Problem consists of finding a shortest walk in a graph that traverses *all* edges and arcs). The authors show that for general precedence relations it is NP-hard to determine a Chinese Postman tour of minimum length. Under strong restrictions on the precedence relation the problem can be solved in time $\mathcal{O}(n^5)$, where n denotes the number of vertices in the input graph.

Online variants of Darp have been studied in [AKR98, FS98]. All of the competitive algorithms from [AKR98, FS98] have to solve offline instances of Darp during their run. The performance of the employed offline algorithm directly affects the competitive ratio of the online algorithm. Thus, the construction of efficient polynomial time (approximation) algorithms for Darp is important to obtain practical, i.e., run time efficient online algorithms.

3. Summary of Results

The results obtained in this paper are summarized in Table 1 and Table 2. Table 1 displays the results for Fifo-Darp obtained in this paper and the known results from literature for Darp. Tabular 2 addresses the problem Penalty-Fifo-Darp which is the extension of Fifo-Darp with start- and stop-penalties discussed in Section 9.

Graph class	Fifo-Darp	Darp
Paths	Polynomial time solvable (Theorem 7.2)	Polynomial time solvable [AK88]
Trees	NP-hard, even on caterpillars (Theorem 6.1) Approximable within 5/3 (Theorem 8.7)	NP-hard, even on caterpillars (Theorem 6.1) Approximable within 5/4 [FG93]
General Graphs	NP-hard Approximable within 9/4 (Corollary 8.6)	NP-hard [FHK78] Approximable within 9/5 [FHK78]

Table 1. Complexity and approximation results for the problems Fifo-Darp and Darp.

Graph class	Penalty-Fifo-Darp
Paths	NP-hard (Lemma 9.4) Approximable within 5/3. (Theorem 9.3)
Trees	NP-hard Approximable within 5/3. (Theorem 9.3)
General Graphs	NP-hard Approximable within 9/4. (Theorem 9.3)

Table 2. Results obtained for the problem Penalty-Fifo-Darp.

4. Basic Observations and Balancing

In this section we illustrate a couple of basic properties of the problem under consideration. We also provide examples for some cases in which intuition might be misleading.

4.1. Simplifying Technical Assumptions.

We first start with some technical assumptions about the input instances depending on the structure of the undirected graph $G[E]$ given in an instance $I = (G = (V, E, A), c, o)$ of FIFO-DARP. While all these assumptions are without loss of generality they greatly simplify the presentation of our algorithms in Sections 7 and 8.

Suppose that $G[E]$ is a tree. Let $v \in V \setminus \{o\}$ be a vertex of degree at most two in $G[E]$ which is neither source nor target of an arc from A. If the degree of v is one, i.e., if v is a leaf, we can remove v and its incident edge without affecting the optimal solution. Similarly, if the degree of v is two, we can replace v and its incident edges by a single edge with cost equal to the sum of the two edges. Thus, for trees we can make the following assumption without loss of generality (cf. [FG93] for DARP on trees):

Assumption 4.1 (Technical assumption for FIFO-DARP on trees). *Each vertex $v \in V$ of degree one or two is either the origin o or incident to at least one arc from A.*

If $G[E]$ is a path, then every vertex has degree one or two. It is easy to see that in this case we can make an even stronger assumption without loss of generality (cf. [AK88] for DARP on paths):

Assumption 4.2 (Technical assumption for FIFO-DARP on paths). *Each vertex $v \in V$ is incident to at least one arc from A.*

We now turn to FIFO-DARP on general graphs.

Assumption 4.3 (Technical assumption for FIFO-DARP on general graphs).

(i) *Each vertex $v \in V$ is incident to at least one arc from A.*
(ii) *$G[E]$ is complete and the cost function c obeys the triangle inequality, i.e., for any edge $[u, v] \in E$ the cost $c(u, v)$ does not exceed the length of a shortest path in $G[E]$ between u and v.*

Note that Assumption 4.3 can be enforced without increasing the value of an optimal solution. If the start vertex o is not incident to any arc from A we can add a new vertex o', a new arc (o, o') and a new edge $[o, o']$, each of cost zero. The new vertex o' is joined by undirected edges to all neighbors v of o. The cost of an edge $[o', v]$ is set to $c(o, v)$. We can then remove vertices which are not source or target of an arc. For every pair u and v of vertices we insert new edges of cost equal to the shortest path in $G[E]$ between u and v. (see also [FHK78] for DARP without precedence constraints).

Assumption 4.3 can not be made without loss of generality for DARP on trees, since removing vertices and later completing the graph as described in general destroys the "tree-property". See also Section 4.3.

4.2. **Balancing.** An important concept for tackling FIFO-DARP on paths and trees is that of *balancing*.

Definition 4.4 (Balancing set). Let $G = (V, E, A)$ be a mixed graph. A multiset $B \subseteq A(E)$ of arcs is called a *balancing set* if in $H = G[A \cup B]$ we have $d^+_H(v) = d^-_H(v)$ for all vertices $v \in H$.

Suppose that $G[E]$ is a tree and that Assumption 4.1 is satisfied. Let $[x, y]$ be an arbitrary edge from E. The removal of $[x, y]$ cuts V into the sets X and $Y := V \setminus X$ with $x \in X$ and $y \in Y$. Any closed walk W in $G = (V, E, A)$ which traverses each arc from A must traverse the cut (X, Y) the same number of times in each direction. Denote by $\phi(X, Y)$ the number of arcs emanating from X, i.e., $\phi(X, Y) := |\{\, r = (x, y) \in A \mid x \in X, y \in Y \,\}|$. Hence, W must must traverse edge $[x, y]$ from x to y at least $b(x, y)$ times, where

$$b(x,y) := \begin{cases} 1 & \text{if } \phi(X,Y) = \phi(Y,X) = 0 \\ \phi(Y,X) - \phi(X,Y) & \text{if } \phi(Y,X) > \phi(X,Y) \\ 0 & \text{otherwise.} \end{cases}$$

The above observation has the following consequence for the equivalent graph augmentation version: If $B \subseteq A(E)$ is a multiset of arcs such that B contains exactly $b(x, y)$ copies of the directed arc (x, y), then there is at least one optimal solution S^* such that $B \subseteq S^*$. This leads to the following lemma which is proved in [AK88, FG93].

Lemma 4.5. *Let $I = (G, c, o)$ be an instance of* DARP *such that $G[E]$ is a tree. Then in time $\mathcal{O}(nm_A)$ one can find a balancing set $B \subseteq A(E)$ such that $B \subseteq S^*$ for some optimal solution S^*.* ☐

Notice also that Lemma 4.5 remains valid even in the presence of FIFO-orders. As is also shown in [AK88, FG93] the time bound of $\mathcal{O}(nm_A)$ can be improved to $\mathcal{O}(n + m_A)$ by allowing balancing arcs to be from $V \times V$ instead of just $A(E)$ (which does not change the problem).[1]

4.3. **Some Notes about Darp on Trees.** An instance of DARP is *balanced* if $G[A]$ is degree-balanced. A balanced instance dissects the directed graph (V, A) consisting of the vertex set V of G and the arcs in A into strongly connected components G_i (each connected component must be strongly connected due to the degree balance). Any component can either be traversed by an Euler tour in A or consists of a single unused vertex in G.

In this case, the *graph of strongly connected components* $\hat{G} = (\hat{V}, \hat{E})$ of G is defined as follows: $\hat{V}$ is the set of all strongly connected components G_i. For two vertices G_i and G_j in $\hat{V}$ we have an edge in $\hat{E}$ if there is an edge $[v_i, v_j] \in E$ from some vertex in $v_i \in G_i$ to some vertex in $v_j \in G_j$. Its cost is set to the shortest edge in E connecting G_i and G_j.

Assume now that we are given a balanced instance of DARP on a tree. Then we are left with the task of connecting a collection of Euler tours by inserting a minimum cost set of new arcs. This task can be accomplished in polynomial

[1] In this case the cost function c is extended from $A(E)$ to $V \times V$ by defining the cost of arc (v, w) to be the length of the shortest path in $G[E]$ from v to w.

time if every component G_i contains either the origin or at least a source or target of an arc from A. In this case, the task can be accomplished by computing a minimum spanning tree (MST) on the complete graph on the vertex set of all strongly connected components; an edge between two strongly connected components has cost equal to the shortest edge between connecting them. Then, for every edge (G_i, G_j) in the MST insert the two arcs (v_i, v_j) and (v_j, v_i), where $v_i \in G_i$ and $v_j \in G_j$ and (v_i, v_j) is the shortest edge between G_i and G_j in G. The collection of arcs so obtained strongly connects the graph and makes it Eulerian. It can be shown that this solution is indeed optimal [AK88] (see also Section 7 for the generalization to FIFO-DARP).

What is different on trees? The point is that we cannot get rid of the unused nodes without destroying the tree-property. Thus, we need to solve a Steiner tree problem rather than an MST problem on the graph of connected components. In some instances where the doubled MST is not optimal, yet the Steiner points to use are canonical (see Figure 2) because, e.g, $\hat{G}$ happens to be a tree, and hence the problem is efficiently solvable. In general, however, we cannot expect this (see Figure 3)—not even on a path. Neither does $\hat{G}$ in general have other, maybe more sophisticated properties that would help to solve the Steiner problem efficiently (see again [FG93] for details).

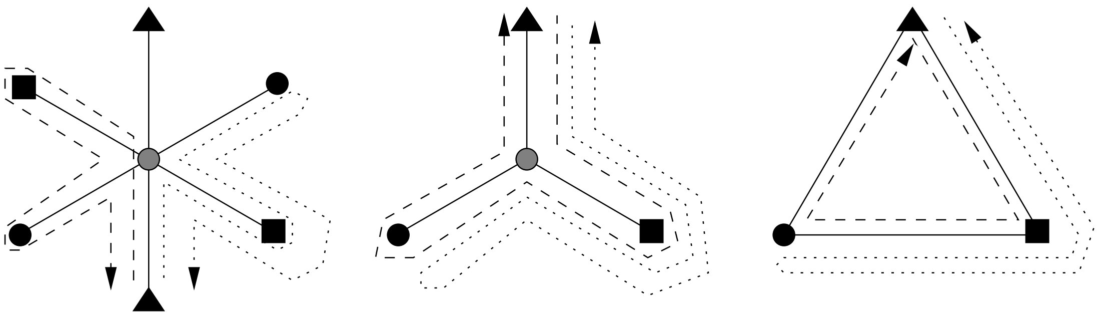

FIGURE 2. The doubled MST solution (dotted) does not equal the optimal tour (dashed) on trees. The given requests are: bring a unit from each symbol to its counterpart; this induces back-and-forth arcs between equally shaped nodes; the grey node is neither start nor end point of any request. From left to right: the instance with four connected components, the graph of strongly connected components $\hat{G}$, and $\hat{G}$ after removal of Steiner points and shortest-path completion.

5. EULER TOURS RESPECTING FIFO-ORDERS

In this section we prove some structural results about Eulerian cycles which respect a given FIFO-order. Notice that it is easy to decide whether a given graph H is $\prec$-Eulerian with start o, provided the restriction of $\prec$ to each set A_v is total: The $\prec$-respecting cycle (if it exists) is uniquely determined and can be

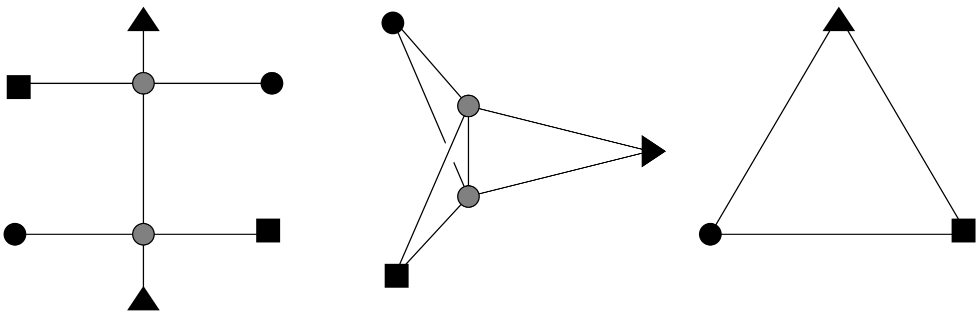

FIGURE 3. A small instance on a tree where $\hat{G}$ is not a tree.

found by a walk through the graph where at each vertex v we always choose among the yet unused arcs from A_v the minimal (with respect to $\prec$). In the sequel we prove a necessary and sufficient condition for a graph to be $\prec$-Eulerian with start at a given vertex.

Let C be an Eulerian cycle starting at o in a directed graph. We define the *set of last arcs* of C, denoted by L, to contain for each vertex $v \in V$ the unique arc emanating from v which is traversed last by C. Observe that L contains a directed spanning tree rooted towards o.

Let $\prec$ be a FIFO-order. We denote the set of maximal elements with respect to $\prec$ by $M_\prec$, that is, $M_\prec := \{\, a \in A : \text{there is no arc } a' \text{ such that } a \prec a' \,\}$.

Definition 5.1 (Possible set of last arcs)**.** Let $H = (V, R)$ be a directed graph and $o \in V$ be a distinguished vertex. A set $L \subseteq R$ is called a *possible set of last arcs*, if it satisfies the following conditions:

(i) $d^+_L(v) = 1$ for all $v \in V$, and
(ii) for each $v \in V$ there is a path from v to o in $H[L]$.

Theorem 5.2. *Let $H = (V, R)$ be a directed Eulerian graph with distinguished vertex $o \in V$ and let $L \subseteq R$ be a possible set of last arcs.*

(i) *There exists an Eulerian cycle C in H such that for each vertex $v \in V$ the (unique) arc from L emanating from v is traversed last at v by C.*
(ii) *Let $\prec$ be any FIFO-order with $L \subseteq M_\prec$. Then there exists a $\prec$-respecting Eulerian cycle with start o in H. This cycle can be found in time $\mathcal{O}(|V| + |R|)$.*

Proof. We first show (i). Color the arcs from L red and the arcs in $R \setminus L$ blue. We claim that by the following procedure we construct an Eulerian cycle C in H with the desired properties. Start with current vertex o. If possible, choose an arbitrary (but yet untraversed) blue arc emanating from the current vertex, otherwise choose the red arc. Traverse the arc, let its target be the new current vertex, and repeat the iteration. Stop, if there is no untraversed arc emanating from the current vertex. Call the resulting path of traversed arcs C. Since H is Eulerian by assumption, for each vertex its in-degree equals its out-degree. Therefore, C must end in the origin o and forms in fact a cycle.

We show that there is no arc in H which is not traversed by C. For a node $v \in V$, let $\operatorname{dist}(v, o)$ be the distance (i.e., the number of arcs) on the shortest path from v to o in the subgraph $H[L]$. We show by induction on $\operatorname{dist}(v,o)$ that all arcs emanating from v are contained in C.

If $\operatorname{dist}(v,o)=0$ then $v=o$. Since our procedure stopped, all arcs emanating from o are contained in C. This proves the induction basis. Assume that the claim holds true for all vertices with distance $t \geq 0$ and let $v \in V$ with $\operatorname{dist}(v,o)=t+1$. Let $a=(v,w)$ be the unique red arc emanating from v. Then $\operatorname{dist}(w,o)=t$ and by the induction hypothesis all arcs emanating from w are contained in C. For $d_H^+(w)=d_H^-(w)$, it follows that all arcs entering w, in particular arc a, are also contained in C. Since red arc a is chosen last by our procedure, all other arcs emanating from v must be contained in C. This completes the induction. Hence, C is actually an Eulerian cycle with the claimed properties.

We proceed to show (ii). Analogously to (i) construct a Eulerian cycle with the sole difference that at each node v we choose the next arc according to the $\prec$-constraint at v. Since by assumption $L \subseteq M_\prec$ this yields a valid $\prec$-respecting Eulerian cycle with start o. □

Corollary 5.3. *Let $H=(V,R)$ be a graph, $o \in V$ and $\prec$ a FIFO-order. Then the following two statements are equivalent:*

1. *H is $\prec$-Eulerian with start o.*
2. *H is Eulerian and the set $M_\prec$ of maximal elements with respect to $\prec$ contains a possible set of last arcs.*

Proof. If H is $\prec$-Eulerian with start o, then the set of last arcs of any $\prec$-respecting Eulerian cycle forms a possible set of last arcs which must be contained in $M_\prec$. Thus Statement 1 implies 2. The other direction is an immediate consequence of part (ii) of Theorem 5.2. □

Observe that Corollary 5.3 in fact implies a polynomial time algorithm for deciding whether a given graph H is $\prec$-Eulerian with start o. Provided H is Eulerian it suffices to check whether the subgraph formed by the arcs from $M_\prec$ contains a directed spanning tree D rooted towards o (which can be done in linear time). Adding to D an arbitrary arc from $A_o \cap M_\prec$ then yields indeed a possible set of last arcs.

6. Hardness Results

Since Fifo-Darp generalizes Darp, it follows from the hardness result in [FG93] that Fifo-Darp is NP-hard even on trees. We show that this hardness continues to hold even if the FIFO-order $\prec$ is total. We can also strengthen the hardness result of [FG93] and show that Darp is hard on caterpillar graphs.

A caterpillar graph is a special case of a tree, consisting of a path, called the *backbone* of the caterpillar, and additional vertices of degree one, called the *feet* of the caterpillar. The edges between vertices on the path and feet are called *hairs.* Notice that caterpillars are trees with maximum degree three.

In our case, the backbone of the caterpillar is the path $G[E]$, the edges between vertices in V and their copies constitute its hairs.

Theorem 6.1. Darp *and* Fifo-Darp *on caterpillars are NP-hard to solve. This result continues to hold, if the transportation jobs are restricted to have* *sources and targets only in the feet of the caterpillar. Furthermore, all hardness results for* FIFO-DARP *remain true if the FIFO-ordering is restricted to be total.*

Proof. We first address the hardness of DARP. To show that DARP on caterpillars is NP-complete, we reduce the Steiner tree problem on bipartite graphs BIPARTITE-STP to it, which is known to be NP-complete [GJ79, Problem ND12]. An instance of BIPARTITE-STP consists of a bipartite graph $H = (X \cup Y, F)$ and a nonnegative number $k \leq |F|$. The problem consists of deciding whether there exists a subtree of H that spans all the vertices in Y and has at most k edges.

In the sequel we assume without loss of generality, that each vertex in Y has degree at least two, since for a vertex $y \in Y$ with degree one, the single edge incident to p has to be included in every tree spanning Y. We also assume without loss of generality that the bipartite graph H is connected. Otherwise, either there is a connected component containing Y, or H can not contain a Steiner tree for the set Y.

We show how to construct an instance $I = (G = (V, E, A), c, o)$ of DARP with $G[E]$ being a caterpillar. We start with the backbone of the caterpillar. We first construct a path of $|X|$ vertices, where for each vertex in X the path contains a copy of this vertex. The weights of the edges on the path are all set to $M := 2|F| + 1$. We then replace each vertex x by a path $P(x)$ of $d_H(x)$ vertices with zero weight on the edges. Here, $d_H(x)$ denotes the degree of x in H. Let Q' denote the vertices on the backbone.

We proceed with the feet and the hairs of the caterpillar. For each vertex $y \in Y$ the set P' of feet contains a set $S(y)$ of $d_H(p)$ vertices. For each edge in F with we add a hair of cost one to the edge set E of the caterpillar. This is done by iteratively choosing an edge $[x, y] \in F$ and then adding a hair between vertices in $P(x)$ and $S(y)$ which are not yet incident to a hair. Notice that there always exist such "free" vertices. Clearly, the graph $G[E]$ constructed this way is a caterpillar graph. The construction is illustrated in Figure 4. The directed arcs are drawn beneath the corresponding vertices to avoid cluttering.

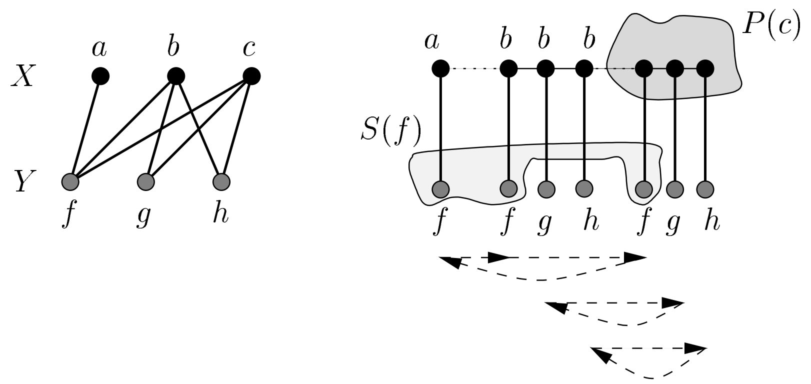

FIGURE 4. Transformation of the Steiner tree problem on a bipartite graph into DARP on a caterpillar in Theorem 6.1

We proceed to construct the transportation jobs A on the caterpillar $G[E]$: Let $y \in Y$ and $S(y) = \{y_1, \dots, y_s\}$ be the set of vertices in G corresponding to y. Then, the set A contains the directed arcs $(y_1, y_2), \dots, (y_{s-1}, y_s), (y_s, y_1)$ which form a simple cycle on $S(y)$. Finally, the origin o of the server is chosen to be the source of an arbitrary arc in A.

Notice that the graph $G[A]$ is by construction degree balanced, i.e., the in-degree of each vertex is equal to its out-degree. Also each set $P(y)$ containing the copies of a vertex $y \in Y$ corresponds to a strongly connected component in $G[A]$. Since $G[A]$ is degree balanced, each strongly connected component is Eulerian and there are no arcs between different components.

Let $K = \sum_{a \in A} c(a)$. We will show, that H contains a Steiner tree with at most k edges if and only if there is a feasible solution to the instance I of DARP with cost at most $K + 2k$, i.e., a multiset $S \subseteq A(E)$ with $c(A \cup S) \leq K + 2k$.

Suppose that T is a Steiner tree in H of at most k edges connecting the vertices in Y. We now describe how to construct a set of arcs S such that $G[A \cup S]$ is Eulerian and contains o and which has cost $c(A \cup S)$ at most $K + 2k$.

Let X_T be the subset of vertices in X, which are spanned by T. For each $x \in X$ we add the antiparallel arcs of zero cost from the path $P(x)$ to S. Notice that these arcs ensure that all the vertices in $P(x)$ will be in the same strongly connected component of $G[A \cup S]$.

For each edge $[x, y] \in T$ we now add two antiparallel arcs between the corresponding vertices in $P(x)$ and and $S(y)$. (Recall that the hairs of the caterpillar correspond to edges in F, such that for each edge $f \in F$ there is a hair connecting copies of the endpoints of f.)

The set S constructed this clearly results in a Eulerian graph $G[A \cup S]$ containing o and has cost $c(S)$ at most $2k$. This shows the first direction.

Assume conversely that S is a feasible solution for the instance I of DARP with cost at most $K + 2k$. Then $G[A \cup S]$ is Eulerian and contains o. The set S can not contain any arc of cost $M = 2|F| + 1$, since otherwise $c(A \cup S) = c(A) + c(S) \geq K + 2|F| + 1 > K + 2k$.

Since $G[A]$ and $G[A \cup S]$ are degree balanced and $S \cap A = \emptyset$, we can decompose the set S into arc disjoint cycles $C_1, \dots, C_p$. Since S contains only (multiple) copies of arcs from $A(E)$ and $G[E]$ is a tree it follows that $r \in S$ implies that the inverse arc r^{-1} must also be contained in S.

We now define a subgraph T of H as follows: If S contains (at least one copy) of an arc between a vertex in $P(x)$ and $S(y)$, then the edge $[x, y]$ is included in T. Since S contains antiparallel pairs of arcs and each arc in S of nonzero cost has cost one, it follows that T has at most $c(S)/2 = k$ edges.

It remains to show that T is connected and spans the vertices in Y. To this end, let y and y' be two arbitrary vertices from Y. Let y_i and y'_j be the corresponding vertices in $S(y)$ and $S(y')$. Since $G[A \cup S]$ is strongly connected, there is a directed path P from y_i to y'_j in $G[A \cup S]$. There are only three types of arcs in P:

1. Arcs from A. These connect the feet $S(\tilde{y})$ corresponding to the same vertex $\tilde{y} \in Y$.

2. Arcs in S with weight zero. These connect vertices on the backbone from the (sub-) path $P(\tilde{x})$ corresponding to the same vertex $\tilde{x} \in X$.
3. Arcs in S of cost one. Such an arc of cost one connects a vertex from a set $P(\tilde{x})$ to a vertex in $S(\tilde{y})$ or vice versa.

For every arc of Type 3 between vertices from in $P(\tilde{x})$ and $S(\tilde{y})$ the subgraph T contains the corresponding undirected edge $[\tilde{x}, \tilde{y}]$. This implies that traversing the edges in T that correspond to arcs in P of Type 3 yields a walk connecting y and y'. This completes the proof of the hardness results for Darp.

To establish the claimed hardness for Fifo-Darp observe that in the above construction each vertex has at most one arc in A emanating from it. Thus, the instance constructed above can also be seen as an instance of Fifo-Darp with total FIFO-ordering. □

7. Fifo-Darp on Paths

In this section we consider Fifo-Darp on paths and show that the problem can be solved in polynomial time. To this end, let $G = (V, E, A)$ be a mixed graph such that $G[E]$ is a path. We assume throughout this section that Assumption 4.2 holds.

Lemma 7.1. *Let $B \cup N$ be the set returned by Algorithm Alg-Path. The set $B \cup N$ is a feasible solution for* Fifo-Darp, *i.e., $G[A \cup B \cup N]$ is $\prec$-Eulerian with start o.*

Proof. In $G[A \cup B]$ each node has in-degree equal to its out-degree. Since N consists of pairs of anti-parallel arcs, the graph $G[A \cup B \cup N]$ is also degree balanced. Since by construction $G[A \cup B \cup N]$ contains a directed spanning tree rooted towards o and $G[A \cup B \cup N]$ is degree balanced it follows that this graph is strongly connected and hence Eulerian.

Input: A mixed graph $G = (V, E, A)$, such that $G[E]$ is a path, a cost function c on E, an initial vertex $o \in V$, and a FIFO-order $\prec$

1 Compute a balancing set $B \subseteq A(E)$ such that $B \subseteq S^*$ for some optimal solution S^*.

2 Let $M_\prec$ be the set of maximal elements with respect to $\prec$.

3 Set $H = G[B \cup M \cup A(E)]$ with cost function c' on the arcs defined by

$$c'(r) = \begin{cases} 0 & \text{if } r \in B \cup M_\prec \\ c(r) & \text{if } r \in A(E) \setminus (B \cup M_\prec) \end{cases}$$

4 Compute a directed spanning tree D rooted towards o of minimum weight $c'(D)$ in $G[B \cup M_\prec \cup A(E)]$.

5 Set $N := \emptyset$. For each directed arc $r \in D$ which is not in $B \cup M_\prec$, add r and its anti-parallel r^{-1} to N.

6 Define $L := D \cup \{r\}$, where r is an arbitrary arc from $A_o \cap (M_\prec \cup B)$
{Notice that such an arc must exist since o is source or target of at least one job and $G[A \cup B]$ is degree-balanced.}

7 Use the method from Theorem 5.2 to find a $\prec$-respecting Eulerian cycle C with start o in $G[A \cup B \cup N]$ such that L is the last set of arcs of C.

8 **return** the set $B \cup N$ and the cycle C.

Algorithm 1: Algorithm Alg-Path for Fifo-Darp on paths.

The set L of arcs determined in Step 6 is clearly a set of possible last arcs. By Theorem 5.2 (ii) there exists indeed a $\prec$-respecting Eulerian cycle with start o in $G[A \cup B \cup N]$. □

Theorem 7.2. *Algorithm* Alg-Path *finds an optimal solution for* FIFO-DARP *on paths.*

Proof. Let S^* be an optimal solution such that $B \subseteq S^*$. By feasibility of S^* the graph $G[A \cup S^*]$ is $\prec$-Eulerian with start o.

We now consider the set $Z := (A \cup S^*) \setminus (A \cup B) = S^* \setminus B$. Since $G[A \cup B]$ and $G[A \cup S^*] = G[A \cup B \cup Z]$ are degree balanced and $Z \cap (A \cup B) = \emptyset$, we can decompose the set Z into arc disjoint cycles $C_1, \dots, C_p$. Since Z contains only (multiple) copies of arcs from $A(E)$ and $G[E]$ is a tree it follows that $r \in Z$ implies that $r^{-1} \in Z$.

Let C be a $\prec$-respecting Eulerian cycle in $G[A \cup S^*]$ and let L be its last set of arcs. Notice that $L \subseteq B \cup M \cup Z$, where the set M is defined in Step 2 of the algorithm. The set L must contain a directed spanning tree D' rooted towards o. We partition D' into the sets $D'_{B \cup M_\prec} := D' \cap (B \cup M_\prec)$ and $D'_Z := D' \cap Z$. Thus, $c'(D'_{B \cup M_\prec}) = 0$ and $c'(D'_Z) = c(D'_Z)$. Since we have seen that for each arc $r \in Z$ also its anti-parallel version $r^{-1} \in Z$ (and D'_Z does not contain a pair of anti-parallel arcs) we get that

$$c(Z) \geq 2c(D'_Z) = 2c'(D'_Z) + 2\underbrace{c'(D'_{B \cup M_\prec})}_{=0} = 2c'(D') \geq 2c'(D). \tag{1}$$

Here, D is the directed spanning tree of minimum weight computed in Step 4. The set N computed in Step 5 has cost

$$c(N) = 2c(D \setminus (B \cup M_\prec)) = 2c'(D \setminus (B \cup M_\prec)) = 2c'(D) \overset{(1)}{\leq} c(Z). \tag{2}$$

Using this result yields that

$$\begin{aligned} c(A \cup B \cup N) &= c(A \cup B) + c(N) = c(A \cup (S^* \setminus Z)) + c(N) \\ &\overset{(2)}{\leq} c(A \cup (S^* \setminus Z)) + c(Z) = c(A \cup S^*). \end{aligned}$$

Thus, $B \cup N$ is an optimal solution as claimed. □

We briefly comment on the running time of Algorithm Alg-Path. Computing a balancing set B can be found in time $\mathcal{O}(nm_A)$ by techniques as shown in [AK88]. As noted before this time bound can be improved to $\mathcal{O}(n + m_A)$ by allowing balancing arcs to be from $V \times V$ instead of just $A(E)$. A rooted spanning tree of minimum weight in a graph with n vertices and m arcs can be computed in time $\mathcal{O}(\min\{m \log n, n^2\})$ by the algorithm from [Tar77] Thus Algorithm Alg-Path can be implemented to run in time $\mathcal{O}(n + m_A + \min\{(m_A + n)\log n, n^2\})$.

8. Approximation Algorithms

8.1. **Algorithms for General Graphs.** In this section we present our approximation algorithm for FIFO-DARP on general graphs. The algorithm uses ideas similar to the ones in [FHK78]. In this section we will assume tacitly that Assumption 4.3 is satisfied.

Input: A mixed graph $G = (V, E, A)$, a cost function c on E, an initial vertex $o \in V$, and a FIFO-order $\prec$

1 Let V_s be the set of vertices which are sources of arcs from A.
2 Compute a complete undirected auxiliary graph U with vertex set V_s. The weight $d(v, w)$ of edge $[v, w]$ is set to be the length of a shortest path in $G[E]$ from v to w.
3 Find an approximately shortest Traveling Salesperson tour P in U starting and ending in o. Let the order in which the vertices of V are visited by P be $v_0 = o, v_1, \dots, v_{|V_s|}, v_{|V_s|+1} = o$.
4 Construct a feasible tour C for FIFO-DARP as follows:
5 Start with the empty tour C.
6 **for** $i := 0, \dots, |V_s|$ **do**
7   Let $a_1, \dots, a_k$ be the arcs from A emanating from vertex v_i. Set $C \leftarrow C + (a_1, p_1, \dots, a_k, p_k)$, where p_j is a shortest path in $G[E]$ from the endpoint of a_j to v_i.
8   Append to C the shortest path in $G[E]$ from v_i to v_{i+1}.
9 **end for**
10 Let S be the multi-set of directed edges used in C which are not contained in A.
11 **return** the set S and the cycle C.

Algorithm 2: TSP-based Approximation Algorithm Alg-TSP for FIFO-DARP.

Our algorithm actually consists of *two* different sub-algorithms, Alg-TSP and Alg-Last-Arcs, which are run both and the best solution is picked. The first sub-algorithm, Alg-TSP, is extremely simple: It computes a shortest tour which visits each vertex from which emanates an arc at least once. Then, it uses this TSP-tour to obtain a feasible solution for FIFO-DARP in the most obvious way. The algorithm is shown in Algorithm 2.

Lemma 8.1. *If in Step 3 a ρ_{TSP}-approximation algorithm for computing a TSP-tour is employed, then algorithm Alg-TSP finds a solution of cost at most $\rho_{\mathrm{TSP}} \cdot OPT + 2c(A)$.*

Proof. Let S^* be an optimum augmenting set and C^* be a $\prec$-respecting Eulerian cycle in $G[A \cup S^*]$ starting at o. Since C^* visits all vertices from V_s, the length of C^* (which equals OPT) is at least that of a shortest TSP-tour on the vertices V_s. Thus, if a ρ_{TSP}-approximation is used, the tour computed in Step 3 will have length at most $\rho_{\mathrm{TSP}} \cdot \mathrm{OPT}$. The additional cost incurred in Step 7 is not greater than $2c(A)$, since each path added has weight not greater than the corresponding arc from A. Hence, the total cost of the cycle C found by algorithm Alg-TSP is bounded from above by $\rho_{\mathrm{TSP}} \cdot \mathrm{OPT} + 2c(A)$ as claimed. ☐

Since the cost of the optimum tour serving all jobs is at least $c(A)$, Lemma 8.1 implies that Alg-TSP is a $(\rho_{\mathrm{TSP}} + 2)$-approximation algorithm for FIFO-DARP. Using Christofides' algorithm [Chr76] we get $\rho_{\mathrm{TSP}} = 3/2$ and thus Alg-TSP provides a 7/2-approximation for FIFO-DARP. In the sequel we will improve this bound by providing a second algorithm and combining this algorithm with Alg-TSP.

Our second algorithm, Alg-Last-Arcs, is based on similar ideas as the algorithm from Section 7 for paths. We first compute a set of balancing arcs B which makes $G[A \cup B]$ degree balanced. Again, we then compute a rooted tree directed towards the origin o of minimum cost, double the arcs which are not yet in $A \cup B$ and add the resulting set N to the solution. Our second algorithm, Alg-Last-Arcs is shown in Algorithm 3.

Input: A mixed graph $G = (V, E, A)$, a cost function c on E, an initial vertex $o \in V$, and a FIFO-order $\prec$

1 Compute a balancing multiset $B \subseteq A(E)$ of minimum cost.
2 Follow steps 2 to 7 of Algorithm Alg-Path to compute a set N of arcs and a $\prec$-respecting Eulerian cycle C with start o.
3 **return** the set $B \cup N$ and the cycle C

Algorithm 3: Algorithm Alg-Last-Arcs "mimicking" the algorithm for paths.

By a proof similar to Lemma 7.1 it follows that the set $B \cup N$ found by Algorithm Alg-Last-Arcs is indeed a feasible solution.

Lemma 8.2. *The balancing set B found in Step 1 of algorithm Alg-Last-Arcs has cost at most $OPT - c(A)$. Step 1 can be accomplished in the time needed for one minimum cost flow computation on a graph with n vertices and $2m_E$ arcs.*

Proof. Let $S^* \subseteq A(E)$ be an optimal solution, i.e., an augmenting (multi-) set of arcs from $A(E)$ with minimum cost. Then the graph $G[A \cup S^*]$ is $\prec$-Eulerian with start o. Thus, in particular, the addition of the arcs from S^* turns G Eulerian, which means that each vertex has an in-degree equal to its out-degree. Thus, the cost $c(S^*) = \mathrm{OPT} - c(A)$ is at least that of a minimum cost set $B \subseteq A(E)$ which achieves the degree balance.

Step 1 can be carried out by performing a minimum cost flow computation in the auxiliary graph $F = (V, A(E))$. A vertex v has charge $d_G^-(v) - d_G^+(v)$ and the cost of sending one unit of flow over arc $r \in A(E)$ equals its cost $c(r)$. We then compute an integral minimum cost flow in F. If the flow on an arc r is $t \in \mathbb{N}$, we add t copies of arc r to the set B. It is easy to see that this yields in fact a balancing set of minimum cost. □

Lemma 8.3. *The cost of the set N computed in Algorithm Alg-Last-Arcs is at most $2(OPT - c(A))$.*

Proof. The proof of the lemma is similar to the one for Theorem 7.2. The major difference is that in general we can not assure that the balancing set B computed in Step 1 is a subset of an optimal solution.

Let S^* be again an optimal augmenting set and L be the set of last arcs of a $\prec$-respecting Eulerian cycle in $G[A \cup S^*]$. We can find a directed spanning tree rooted towards o in L. The only arcs from A that L can contain are those from the set $M_\prec$. Thus $L \setminus (A \cup B) = L \setminus (M_\prec \cup B)$. Similar to Theorem 7.2 we can now conclude that

$$\mathrm{OPT} - c(A) = c(S^*) \geq c(L \setminus (A \cup B)) = c(L \setminus (M_\prec \cup B)) = c'(L) \geq \frac{c(N)}{2}.$$

This shows the claim. □

Corollary 8.4. *Algorithm Alg-Last-Arcs finds a solution of cost at most $3 \cdot OPT - 2c(A)$.*

Proof. By Lemma 8.2, $c(A \cup B) \leq \mathrm{OPT}$. Lemma 8.3 establishes that $c(N) \leq 2 \cdot \mathrm{OPT} - 2c(A)$. Thus $c(A \cup B \cup N) \leq 3 \cdot \mathrm{OPT} - 2c(A)$ as claimed. □

We are now ready to combine our algorithms into one with an improved performance guarantee. The combined algorithm Alg-Combine simply runs both algorithms and picks the better solution.

Theorem 8.5. *Algorithm Alg-Combine has a performance of* $\frac{\rho_{\mathrm{TSP}}+3}{2}$.

Proof. Let $\beta := \frac{4}{3-\rho_{\mathrm{TSP}}}$. If $\mathrm{OPT} \leq \beta c(A)$, then the solution returned by Alg-TSP has cost at most

$$(\rho_{\mathrm{TSP}} + 2/\beta)\,\mathrm{OPT} = \left(\rho_{\mathrm{TSP}} + 2\,\frac{3-\rho_{\mathrm{TSP}}}{4}\right)\mathrm{OPT} = \frac{\rho_{\mathrm{TSP}}+3}{2}\cdot \mathrm{OPT}.$$

If $\mathrm{OPT} > \beta c(A)$, then the cost of the solution found by Alg-Last-Arcs is bounded from above by

$$(3 - 2/\beta)\,\mathrm{OPT} = \left(3 - 2\,\frac{3-\rho_{\mathrm{TSP}}}{4}\right)\mathrm{OPT} = \frac{\rho_{\mathrm{TSP}}+3}{2}\cdot \mathrm{OPT}.$$

This shows the claim of the theorem. □

Using Christofides' algorithm [Chr76] with $\rho_{\mathrm{TSP}} = 3/2$ results in a performance guarantee of $3/4 + 3/2 = 9/4$ for algorithm Alg-Combine.

Corollary 8.6. *There is an approximation algorithm for* FIFO-DARP *with performance* $9/4$*. This algorithm can be implemented to run in time* $\mathcal{O}(\max\{n^3 + m_A m_E + m_A n \log n, m_E^2 \log n + m_E n \log^2 n\})$.

Proof. The performance has already been proved. The running time of Algorithm Alg-TSP is dominated by that of Christofides' algorithm, which can be implemented to run in time $\mathcal{O}(n^3)$, and the time needed for the addition of the paths in Step 7 which can be done in total time $\mathcal{O}(m_A m_E + m_A n \log n)$. The running time of Alg-Last-Arcs is dominated by the minimum cost flow computation which can be accomplished in time $\mathcal{O}(m_E^2 \log n + m_E n \log^2 n)$ by using Orlin's enhanced capacity scaling algorithm [AMO93]. □

8.2. **Improved Performance on Trees.** For graph classes where the TSP can be approximated within a factor better than $3/2$, the performance improves over the one stated in Corollary 8.6. In particular, for trees where the TSP can be solved in polynomial time Theorem 8.5 already implies a 2-approximation algorithm. However, we can still improve this performance guarantee.

Theorem 8.7. *There exists a polynomial time approximation algorithm for* FIFO-DARP *on trees with performance* $5/3$*. This algorithm can be implemented to run in time* $\mathcal{O}(nm_A + n^2 \log n)$.

Proof. Our algorithm for trees uses a modified version of Alg-Last-Arcs. We defer removal of the vertices in V which are neither start nor endpoint of an arc from A and the completion of G via shortest paths until after the (modified) balancing step. The balancing step Step 1 of Alg-Last-Arcs is modified so that we find a balancing subset $B \subseteq S^*$ as in Lemma 4.5. After the balancing we remove all vertices which are not incident to the arcs in $A \cup B$ and continue with Alg-Last-Arcs from Step 2 on.

Let $I = (G = (V, E, A), c, o, \prec)$ be the original instance given such that $G[E]$ is a tree. We can consider the instance $I' = (G = (V, E, A \cup B), c, o, \prec)$ of

FIFO-DARP (still on a tree) which results from adding the balancing arcs B as new transportation jobs. Since any feasible solution to I will have to use the arcs from B anyway (cf. Lemma 4.5), we get that $\mathrm{OPT}(I) = \mathrm{OPT}(I')$.

Now look at the instance I'' of FIFO-DARP which is obtained by removing vertices and completing G along shortest paths as in our algorithm. It is easy to see that $\mathrm{OPT}(I'') = \mathrm{OPT}(I')$. Notice also that we can transform any feasible solution to I'' to a feasible solution to I' (by replacing arcs not in $A(E)$ by shortest paths). Let S^* and S'' be optimal solutions for I and I'', respectively. Define $Z := S^* \setminus B$ and $Z'' := S'' \setminus B$. Since $\mathrm{OPT}(I) = c(A \cup B) + c(Z) = \mathrm{OPT}(I'') = c(A \cup B) + c(Z'')$, we have that $c(Z) = c(Z'')$.

Let $A \cup B \cup N$ be the solution found by the modified version of Alg-Last-Arcs. Then, using the arguments of Lemma 8.3 we get that

$$\begin{aligned} c(A \cup B \cup N) &= c(A \cup B) + c(N) = c(S^*) - c(Z) + c(N) \\ &\leq c(S^*) - c(Z) + 2c(Z'') \leq c(S^*) + c(Z) \\ &= 2 \cdot \mathrm{OPT}(I) - c(A). \end{aligned}$$

As noted before, Alg-TSP finds a solution of cost at most $\mathrm{OPT} + 2c(A)$, since we can solve the TSP on the tree $G[E]$ in polynomial time. We can estimate the cost of the best of the two solutions returned by Alg-TSP and the modified Alg-Last-Arcs by the techniques from Theorem 8.5 where this time $\beta = 3$. This yields a performance of 5/3 as claimed.

The time bound for the algorithm is derived as follows: We can solve the TSP on the metric space induced by $G[E]$ in time $\mathcal{O}(n)$. We then root the tree $G[E]$ at an arbitrary vertex. With $\mathcal{O}(n)$ preprocessing time, the least common ancestor of any pair of vertices can be found in constant time (see [HT84, SV88]). Thus, we can implement Alg-TSP in such a way that the invocations of Step 7 take total time $\mathcal{O}(nm_A)$. This means that Alg-TSP can be implemented to run in time $\mathcal{O}(nm_A)$.

The balancing in the modified version of Alg-Last-Arcs can be accomplished in time $\mathcal{O}(n + m_A)$. Completion of the graph by computing all-pairs shortest paths can be done in time $\mathcal{O}(nm_E + n^2 \log n) = \mathcal{O}(n^2 \log n)$ [CLR90, AMO93]. All other steps can be carried out in time $\mathcal{O}(n^2)$ where again the algorithm from [Tar77] is employed for computing a minimum weight directed spanning tree. □

9. FIFO-DARP WITH START AND STOP PENALTIES

In this section we show how to incorporate additional start- and stop-penalties into the problem FIFO-DARP. In the Dial-a-Ride-Problem with penalties, short PENALTY-FIFO-DARP, we are given additional penalty functions p^+ and p^- on the set of vertices, where $p^+(v)$ is the time penalty for starting from a vertex and $p^-(v)$ is the penalty for stopping at a vertex. The objective is to find a closed walk serving all requests, such that the cost of the walk plus the cost of starting and stopping is minimized.

To formulate PENALTY-FIFO-DARP in a meaningful way as a graph augmentation problem, we have to allow augmenting arcs from $V \times V$ and not just from $A(E)$, since each arc corresponds to a move and incurs a start and stop penalty. The cost function $c\colon E \to \mathbb{R}_{\geq 0}$ is extended by defining the cost of arc (v, w) to be the length of a shortest path from v to w in $G[E]$.

Definition 9.1 (Graph augmentation version of PENALTY-FIFO-DARP)**.** An instance of PENALTY-FIFO-DARP consists of the same input as for FIFO-DARP together with additional penalty functions $p^+, p^- : V \to \mathbb{R}_{\geq 0}$ on the set of vertices V. The objective is to find a multiset of arcs S from $V \times V$ minimizing the weight

$$c(A \cup S) + \sum_{u \in U^+} d^+(u)p^+(u) + \sum_{u \in U^-} d^-(u)p^-(u)$$

such that $G[A \cup S]$ is $\prec$-Eulerian with start o. Here, U^+ is the set of sources of arcs in $A \cup S$ and U^- is the set of endpoints of arcs in $A \cup S$.

In the sequel we show that an instance $I = (G = (V, E, A), c, o, \prec, p^-, p^+)$ of PENALTY-FIFO-DARP can be transformed into an equivalent instance of FIFO-DARP $I' = (G' = (V', E', A'), c', o', \prec')$ on a slightly larger graph.

The transformation is accomplished as follows: For each vertex $v \in V$ we add both v and a new vertex $v^{(\pm)}$ to V'. Vertex $v^{(\pm)}$ is used to model starting or stopping at vertex v. The set E' consists of the edges in E and an additional edge e_v between v and $v^{(\pm)}$ for each vertex $v \in V$. The cost of the new edges is $c'(e_v) = 1/2(p^+(v) + p^-(v))$. The cost function c' coincides with c on the set E. For each arc $a = (u, v) \in A$ we add an arc $a' = (u^{(\pm)}, v^{(\pm)})$ to A (the arcs in A are not contained in A'). The partial order on the set $A_{u^{(\pm)}}$ is induced in the obvious way by that on A_u. Finally, the start vertex o' equals o.

Lemma 9.2. *Let $I = (G, c, o, \prec, p^+, p^-)$ be an instance of* PENALTY-FIFO-DARP *and $I' = (G', \prec, c', o')$ be the instance of* DARP *constructed by the above method. Then, I and I' are equivalent in the following sense: Any feasible solution for I' can be transformed into a feasible solution for I of the same cost and vice versa. This transformation can be accomplished in polynomial time.*

Proof. Let S' be a valid solution for problem instance I' of FIFO-DARP where S' is an augmenting set of arcs. Let C' be a $\prec$-respecting Eulerian cycle in $G'[A' \cup S']$ with start o'.

We first construct an auxiliary set M of arcs by traversing C' and replacing all chains of arcs from S' with a single arc from the start vertex of the chain to its end vertex. Notice that all endpoints of arcs in M are contained in $V' \setminus V$. We now construct a solution S by replacing each arc $(u^{(\pm)}, v^{(\pm)})$ by (u, v). It is easy to see that S is in fact a valid solution for I of cost equal to that of S'.

Conversely, let S be a feasible solution for I. We can construct a solution S' for I' with equal cost by adding to S' for each arc (u, v) in S the arc $(u^{(\pm)}, v^{(\pm)})$.

The time bound is obvious from the construction. □

It follows from the construction that if $G[E]$ is a tree then $G'[E']$ is also a tree. Thus, the last lemma implies that approximation results for FIFO-DARP on trees can be applied directly to PENALTY-FIFO-DARP on trees. Similarly, approximation results for general graphs carry over immediately. Hence, we obtain the following result:

Theorem 9.3. *The problem* PENALTY-FIFO-DARP *can be approximated on trees with performance* 5/3 *and with performance* 9/4 *on general graphs.* ☐

However, transforming an instance of PENALTY-FIFO-DARP where $G[E]$ is a path yields an instance of FIFO-DARP where $G'[E']$ is a caterpillar graph. This seems unfortunate, since we know from Theorem 6.1 that FIFO-DARP is NP-hard to solve on caterpillars. Is there a better transformation? More general, is PENALTY-FIFO-DARP on paths still polynomial time solvable?

The caterpillar constructed in the proof of Theorem 6.1 has the property that jobs have sources and targets only in the feet of the caterpillar. Actually every instance of FIFO-DARP on caterpillars with these properties can be transformed into an equivalent instance of PENALTY-FIFO-DARP on a path: Let f be a foot and v be its unique adjacent vertex on the backbone. We replace all arcs from A which are incident with f by corresponding arcs with source or target v. We then remove foot f. The start- and stop-penalty on v are set to the length $c(f, v)$ of the hair between v and the foot f. It follows by arguments similar to those given in Lemma 9.2 that the constructed instance of PENALTY-FIFO-DARP on the path (which corresponds to the former backbone) is in fact an equivalent instance to the instance of FIFO-DARP on the caterpillar. Thus, we obtain the following result which contrasts with the polynomial solvability of FIFO-DARP on paths:

Lemma 9.4. PENALTY-FIFO-DARP *on paths is NP-hard to solve.* ☐

10. Concluding Remarks

We have presented a natural extension of a "Dial-a-Ride-Problem", which was originally motivated by the performance analysis of a large distribution center of Herlitz AG, Berlin [AG+98]. We have shown that even in the presence of FIFO-precedence constraints for the transportation jobs the problem can be solved in polynomial time on paths. On trees, however, the problem is NP-hard. Our approximation algorithms for general graphs and trees with performance 9/4 and 5/3, respectively, compare well to the performances of 9/5 [FHK78] and 5/3 [FG93] obtained for DARP without precedence constraints.

References

[AG+98] N. Ascheuer, M. Grötschel, S. O. Krumke, and J. Rambau, *Combinatorial online optimization*, Proceedings of the International Conference of Operations Research (OR'98), Springer, 1998, pp. 21–37.

[AK88] M. J. Atallah and S. R. Kosaraju, *Efficient solutions to some transportation problems with applications to minimizing robot arm travel*, SIAM Journal on Computing **17** (1988), no. 5, 849–869.

[AKR98] N. Ascheuer, S. O. Krumke, and J. Rambau, *Competitive scheduling of elevators*, Preprint SC 98-34, Konrad-Zuse-Zentrum für Informationstechnik Berlin, November 1998.

[AMO93] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Networks flows*, Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[Chr76] N. Christofides, *Worst-case analysis of a new heuristic for the traveling salesman problem*, Tech. report, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, 1976.

[CLR90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to algorithms*, MIT Press, 1990.

[DST87] M. Dror, H. Stern, and P. Trudeau, *Postman tour on a graph with precedence relation on the arcs*, Networks **17** (1987), 283–294.

[FG92] G. N. Frederickson and D. J. Guan, *Preemptive ensemble motion planning on a tree*, SIAM Journal on Computing **21** (1992), no. 6, 1130–1152.

[FG93] G. N. Frederickson and D. J. Guan, *Nonpreemptive ensemble motion planning on a tree*, Journal of Algorithsm **15** (1993), no. 1, 29–60.

[FHK78] G. N. Frederickson, M. S. Hecht, and C. E. Kim, *Approximation algorithms for some routing problems*, SIAM Journal on Computing **7** (1978), no. 2, 178–193.

[FS98] E. Feuerstein and L. Stougie, *On-line single server dial-a-ride problems*, Manuscript, submitted for publication, 1998.

[GJ79] M. R. Garey and D. S. Johnson, *Computers and intractability (a guide to the theory of NP-completeness)*, W.H. Freeman and Company, New York, 1979.

[Gua98] D. J. Guan, *Routing a vehicle of capacity greater than one*, Discrete Applied Mathematics **81** (1998), no. 1, 41–57.

[HT84] D. Harel and R. E. Tarjan, *Fast algorithms for finding nearest common ancestors*, SIAM Journal on Computing **13** (1984), no. 2, 338–355.

[SV88] B. Schieber and U. Vishkin, *On finding lowest common ancestors: Simplification and parallelization*, SIAM Journal on Computing **17** (1988), no. 6, 1253–1262.

[Tar77] R. E. Tarjan, *Finding optimum branchings*, Networks **7** (1977), 25–35.

Dietrich Hauptmeier, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Department Optimization, Takustr. 7, D-14195 Berlin-Dahlem, Germany.
E-mail address: `hauptmeier@zib.de`

Sven O. Krumke, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Department Optimization, Takustr. 7, D-14195 Berlin-Dahlem, Germany.
E-mail address: `krumke@zib.de`
URL: `http://www.zib.de/krumke`

Jörg Rambau, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Department Optimization, Takustr. 7, D-14195 Berlin-Dahlem, Germany.
E-mail address: `rambau@zib.de`
URL: `http://www.zib.de/rambau`

Hans-Christoph Wirth, Department of Computer Science, University of Würzburg, Am Hubland, 97074 Würzburg, Germany
E-mail address: `wirth@informatik.uni-wuerzburg.de`
URL: `http://www-info1.informatik.uni-wuerzburg.de/mitarbeiter/wirth`