ELSEVIER
Discrete Applied Mathematics 113 (2001) 87–107
DISCRETE APPLIED MATHEMATICS

Euler is standing in line dial-a-ride problems with precedence-constraints

D. Hauptmeier[a], S.O. Krumke[a, *,1], J. Rambau[a], H.-C. Wirth[b]

[a] *Department Optimization, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Takustr. 7, 14195 Berlin-Dahlem, Germany*
[b] *Department of Computer Science, University of Würzburg, Am Hubland, 97074 Würzburg, Germany*

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 approximation algorithms for general graphs and trees with performances of 9/4 and 5/3, respectively.

Keywords: Vehicle routing; Elevator system; Eulerian cycle; Approximation algorithms

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 which we call S-DARP is motivated by applications in which first-in-first-out waiting lines are present at the sources of the

* Corresponding author.
E-mail addresses: hauptmeier@zib.de (D. Hauptmeier), krumke@zib.de (S.O. Krumke), rambau@zib.de (J. Rambau), wirth@informatik.uni-wuerzburg.de (H.-C. Wirth).

[1] Research supported by the German Science Foundation (DFG, grant Gr 883/5-2).

0166-218X/01/$ - see front matter
PII: S0166-218X(00)00390-5

transportation jobs. In this case, jobs can be served only according to their order in the line. Examples with first-in-first-out 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.

This paper is organized as follows. In Section 2 we formally state the problem S-DARP. We also show that S-DARP can be equivalently formulated as a graph augmentation problem. This key observation will be used to design our algorithms. In Section 4 we prove structural facts about Eulerian cycles in a graph that respect a given "source-order" on the arcs. (A formal definition of source-orders appears in Section 2.2.)

Section 5 contains a polynomial algorithm for S-DARP when restricted to paths. In Section 6 we present an approximation algorithm for general graphs with performance of $(\rho_{\mathrm{T}_{SP}} + 3)/2$, where $\rho_{\mathrm{T}_{SP}}$ is the performance of the best approximation for the TSP with triangle inequality. An improved algorithm for trees with performance 5/3 is given in Section 7. Section 8 is dedicated to hardness results. Section 9 briefly discusses the extension of our results to include start and stop penalties.

2. Preliminaries and problem formulations

A *multiset* X over a ground set U, denoted by $X \sqsubset U$, can be defined as a mapping $X : U \to \mathbb{N}$, where for $u \in U$ the number $X(u)$ denotes the multiplicity of u in X. We write $u \in X$ if $X(u) \geqslant 1$. Any (standard) set can be viewed as a multiset with elements of multiplicity 0 and 1. If $Y \sqsubset U$ then $X \sqsubset Y$ denotes a multiset over the ground set $\{u \in U : Y(u) > 0\}$. If $X \sqsubset U$ and $Y \sqsubset U$ are multisets over the same ground set U, then we denote by $X + Y$ their *multiset union*, by $X - Y$ their *multiset difference* and by $X \cap Y$ their *multiset intersection*, defined for $u \in U$ by

$$(X + Y)(u) = X(u) + Y(u),$$

$$(X - Y)(u) = \max\{X(u) - Y(u), 0\},$$

$$(X \cap Y)(u) = \min\{X(u), Y(u)\}.$$

The multiset $X \sqsubset U$ is a subset of the multiset $Y \sqsubset U$, denoted by $X \subseteq Y$, if $X(u) \leqslant Y(u)$ for all $u \in U$. For a weight function $c : U \to \mathbb{R}$ the weight of a multiset $X \sqsubset U$ is defined by $c(X) := \sum_{u \in U} c(u)X(u)$. We denote the cardinality of a multiset $X \sqsubset U$ by $|X| := \sum_{u \in U} X(u)$.

A *mixed graph* $G = (V, E, A)$ consists of a set V of vertices, a set E of undirected edges without parallels, and a multiset 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 $v \in V$ we let A_v be the set of arcs in A emanating from v. For edge set E, denote by

$$\vec{E} := \{(u, v), (v, u) : [u, v] \in E\} \tag{1}$$

the set of arcs which contains for each undirected edge $e \in E$ a pair of anti-parallel arcs between the endpoints of e.

If $X \sqsubset E + A$, 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 incident vertices. A subgraph of G *induced by vertex set* $X \subseteq V$ is a subgraph with node set X and containing all those edges and arcs from G which have both endpoints in X. Throughout the paper we assume that $G[E]$ is connected and for each arc from A contains both endpoints. 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 $X \sqsubset A$, 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* W in the mixed graph $G = (V, E, A)$ is an alternating sequence of vertices and edges/arcs $W = (v_1, x_1, v_2, \ldots, x_k, v_{k+1} = v_1)$ where $v_i \in V$ and $x_i \in E + A$ for $i = 1, \ldots, k$ such that for any i either x_i is an undirected edge $[v_i, v_{i+1}]$ between v_i and v_{i+1} or a directed arc (v_i, v_{i+1}) from v_i to v_{i+1}. Notice that we allow the walk w to visit vertices, edges and arcs multiple times. For a cost function $c : E + A \to \mathbb{R}_{\geqslant 0}$ the *cost* of the walk W is given by $c(W) := \sum_{i=1}^{k} c(x_i)$.

A *directed spanning tree rooted towards* $o \in V$ is a subgraph 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 $\mathsf{A}(I) \leqslant \rho\,\mathrm{OPT}(I)$.

2.1. Basic problem

In the "Dial-a-Ride Problem" DARP we are given a finite transportation network and a finite set of transportation jobs. Each job specifies the source and target location which are both part of the network. A server which can carry at most one object at a time can move on the transportation network to process the transportation requests. The problem DARP consists of finding a shortest transportation for the jobs starting and ending at a designated start location.

We model the transportation network by an edge weighted undirected graph. Each job request is modelled by an arc from its source node to its target node. The length of the arc is adjusted to reflect the length of a shortest path in the network connecting its endpoints. Then a closed walk in the resulting mixed graph which traverses each arc corresponds to a transportation for all the jobs in the transportation network. More formally, we define DARP as follows:

Definition 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 $c: E \to \mathbb{R}_{\geqslant 0}$.

The weight function c is extended to A by defining for each arc $a \in A$, $a=(v,w)$, its cost $c(a)$ to be the length of a shortest path from v to w in $G[E]$.

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

It turns out that for the purpose of stating algorithms in a more convenient way, it is helpful to use an equivalent formulation of DARP as a *graph augmentation* problem (cf. [5]). To this end consider the arc set $\vec{E}$ defined in (1). Let the cost of each arc in $\vec{E}$ equal the cost of the corresponding edge in E.

With these definitions, we formulate a graph augmentation problem: Given mixed graph $G=(V,E,A)$, origin $o \in V$, and cost function $c: E \to \mathbb{R}_{\geqslant 0}$, extend c to $A+\vec{E}$ as described in Definition 1 and the previous paragraph. Then find a multiset R, $R \sqsubset \vec{E}$, of minimum cost such that graph $G[A+R]$ is Eulerian and contains o. We argue that this problem is an equivalent formulation of DARP.

Let W be a feasible solution for DARP as stated in Definition 1, that is, a closed walk that starts in o and traverses each arc in A. Construct a multiset $R \sqsubset \vec{E}$ of arcs in the following way: Traverse edges and arcs along W. For each time an undirected edge $e \in E$, $e=[u,v]$, is traversed from u to v, add a copy of the directed arc (u,v) to multiset R. Then graph $G[A+R]$ contains o, and since W defines a cycle, graph $G[A+R]$ must be Eulerian.

Conversely, let $R \sqsubset \vec{E}$ be a multiset of arcs such that $G[A+R]$ is Eulerian and includes the origin o. Construct a walk W as follows: Traverse an Eulerian cycle C in $G[A+R]$ starting in o. If the current arc r from C is in A then add r to walk W, otherwise add the undirected edge corresponding to r. By this construction, W is a closed walk in G traversing each arc from A and including o. In both cases we have $c(W)=c(A+R)$, i.e., the cost of walk W equals the cost of multiset R plus cost of arc set A.

2.2. Precedence constraints

In real applications of DARP there are often additional constraints on the order of the execution of transportation requests. This can be modelled by introducing a partial order $\prec$ on the set of arcs. A feasible walk is then required to satisfy the condition that, whenever $a \prec b$, then a must be traversed before b by the walk.

In some transportation networks there is a "waiting pool" at each node where transportation requests originate. Each of the pools constraints the order of execution of requests starting from this node while requests starting from other nodes are not affected. For instance there might be waiting pools with first-in first-out queues or waiting stacks (last-in first-out). This motivates the definition of a *source-order*, which is a partial order satisfying: $a \prec b$ implies that a and b share the same source node. If a

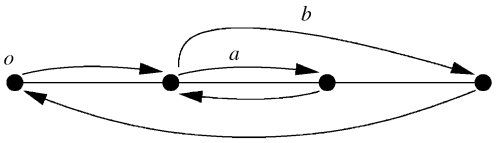

Fig. 1. Precedence constraint $b \prec a$ increases the cost.

source-order $\prec$ satisfies: a and b share the same source node implies $a \prec b \vee b \prec a$, then $\prec$ is called a *total source-order*.

This leads to problem S-DARP (shorthand for "DARP with source-orders") which is the main focus of this paper. An instance of S-DARP consists of an instance of DARP, together with a source-order $\prec$ on the arc set A. The goal is to find a closed walk satisfying the requirements specified in Definition 1 and the precedence constraint given by $\prec$.

Fig. 1 shows an example where an optimal source-order respecting transportation is strictly longer than the optimal transportation neglecting the precedences. If no constraints have to be obeyed, then the jobs can be served traversing only along the arcs. If the constraint $b \prec a$ must be obeyed then two additional empty moves along undirected edges are necessary.

For the sake of presentation it will be useful to formulate S-DARP as a graph augmentation problem. We need some additional notations:

Definition 2 (*$\prec$-respecting Eulerian cycle, $\prec$-Eulerian*)**.** Let $H = (V,R)$ be a directed graph, $\prec$ be a source-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.

Definition 3 (*Graph augmentation version of* S-DARP)**.** An instance of the problem S-DARP consists of the same input as for DARP and additionally a source-order $\prec$ on the arc set A. The goal is to find a multiset S of arcs from $\vec{E}$ minimizing the weight $c(A+S)$ such that $G[A+S]$ is $\prec$-Eulerian with start o, and to determine $a \prec$-respecting Eulerian cycle in $G[A+S]$.

2.3. Related work

The problem DARP is also known as the Stacker–Crane-Problem. In [11] it is shown that the problem is NP-hard even on trees, i.e., if the graph $G[E]$ is a tree. In [12] 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 [11]. On paths DARP can be solved in polynomial time [5].

The extension of DARP where a vehicle of capacity $C > 1$ is used to serve the transportation jobs has been addressed in [14,6]. For capacity $C > 1$ the problem becomes NP-hard even on paths. In [6] an approximation algorithm for the single server dial-a-ride problem with performance $\mathcal{O}(\sqrt{C}\log n \log\log n)$ was given, where C denotes the capacity of the server and n denotes the number of vertices in the graph.

Precedence constraints have been studied in the case of Chinese Postman tours in [9]. (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 investigated in [3,4,10]. All of the known competitive algorithms 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. If a ρ-approximation algorithm for DARP is given, the "Smartstart"-strategy presented in [4] yields a polynomial time c_ρ-competitive algorithm for online-DARP with competitive ratio $c_\rho = \frac{1}{4}(4\rho + 1 + \sqrt{1 + 8\rho})$.

3. Basic observations and balancing

Throughout the paper we will use $S^* \sqsubset \vec{E}$ to denote an optimal solution (augmentation set) and OPT:$=c(A + S^*)$ to denote its cost. We first start with some technical assumptions about input instances $(G = (V, E, A), c, o, \prec)$ of S-DARP. While all these assumptions are without loss of generality they greatly simplify the presentation of our algorithms.

Assumption 4 (Technical assumption for S-DARP on trees)**.** *Each vertex of degree one or two in $G[E]$ is either the origin o or incident to at least one arc from A.*

The above assumption is indeed no restriction: Let $v \neq o$ be not adjacent to any arc in A. If v is of degree two, replace the two adjacent edges, say $[v, u]$ and $[v, w]$, by the single edge $[u, w]$ of cost equal to the sum of the two edges. If v is a leaf, it can be removed without affecting the optimal solution (cf. [11] for DARP on trees).

If $G[E]$ is a path it is easy to see that we can make an even stronger assumption without loss of generality (cf. [5] for DARP on paths):

Assumption 5 (Technical assumption for S-DARP on paths)**.** *Each vertex $v \in V$ is incident to at least one arc from A.*

We now turn to S-DARP on general graphs.

Assumption 6 (Technical assumption for S-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* $e \in E$, $e = [u, v]$, *the cost* $c(e)$ *does not exceed the length of a shortest path in* $G[E]$ *between* u *and* v.

Assumption 6 can be enforced without increasing the value of an optimal solution. If the start vertex o is not incident to any arc from A, insert a new vertex o' joined by arc (o, o') and edge $[o, o']$ to the start vertex, each of cost zero. To satisfy the triangle inequality, for every pair u, v of vertices add a new edge $[u, v]$ of cost equal to the shortest path in $G[E]$ between u and v.

Afterwards each bundle of parallel edges can be replaced by retaining the cheapest edge of the bundle, and vertices which are not incident to an arc can be removed safely (cf. [12] for DARP).

However, Assumption 6 cannot be made without loss of generality for S-DARP on trees, since the suggested modification of the graph destroys the "tree-property".

A necessary condition for a graph to be Eulerian is that for each node its in-degree equals its out-degree. It turns out to be helpful for solving S-DARP to search for augmenting sets which guarantee the resulting graph to be *balanced* in that way.

Definition 7 (*Balancing set*)**.** Let $G = (V, E, A)$ be a mixed graph. A multiset $B \sqsubset \vec{E}$ of arcs is called a *balancing set* if in $H = G[A + B]$ we have $d_H^+(v) = d_H^-(v)$ for all vertices v of H.

Suppose that $G[E]$ is a tree and that Assumption 4 is satisfied. For a partition $V = X \cup Y$ of the vertex set define the cut $(X : Y)$ to consist of all edges and arcs from $E + A$ with one endpoint in X and the other one in Y. Any edge $[x, y] \in E$ defines a partition $V = X \cup Y$ of the node set, where X and Y are defined by the connected components after removing the edge $[x, y]$ from the tree. Obviously, the cut $(X : Y)$ must be traversed by any closed walk W the same number of times in each direction. Denote by $\phi(X, Y) := |\{(x, y) \in A \mid x \in X \wedge y \in Y\}|$ the number of arcs emanating from X. Hence, if W traverses all arcs from A, it must traverse edge $[x, y]$ from x to y at least $b(x, y)$ times, where

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

This observation has the following consequence for the graph augmentation version: If $B \sqsubset \vec{E}$ is a multiset of arcs such that $B((x, y)) = b(x, y)$, that is, B contains $b(x, y)$ copies of the directed arc (x, y), then there is at least one optimal solution S^* such that $B \subseteq S^*$. This yields the following lemma which is proved in [5,11].

Lemma 8. *Let* (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* $B \sqsubset \vec{E}$ *such that* $B \subseteq S^*$ *for some optimal solution* S^*.

Notice that Lemma 8 remains valid even in the presence of source-orders. As is also shown in [5,11] 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 $\vec{E}$ (which does not change the problem: the cost function c is extended from $\vec{E}$ to $V \times V$ by the length of shortest paths).

4. Euler cycles respecting source-orders

Let C be an Eulerian cycle starting at o in a connected directed graph. We define the *set of last arcs* of C, denoted by L_C, to contain for each vertex $v \in V$ the unique arc emanating from v which is traversed last by C. One can observe that L_C consists of a directed spanning tree rooted towards o plus one single arc emanating from o. In the following we will examine the situation for source-order respecting Eulerian cycles.

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

Definition 9 (*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 $H[L]$.

We remark that this definition is equivalent to the following: a set L is a possible set of last arcs, if $H[L]$ is a directed spanning tree rooted towards o, plus one arbitrary arc emanating from o. The following theorem justifies the nomenclature "possible set of last arcs":

Theorem 10. *Let $H=(V,R)$ be a directed Eulerian graph with distinguished vertex $o \in V$ and let $\prec$ be a source-order with maximal elements $M_\prec$. Suppose that a possible set L of last arcs satisfies $L \subseteq M_\prec$.*

Then there exists an $\prec$-respecting Eulerian cycle C with start o in H such that $L_C = L$, i.e., such that L is the set of last arcs of C. This cycle can be found in time $\mathcal{O}(|V| + |R|)$.

Proof. 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. As long as there is a blue untraversed arc emanating from the current vertex, choose one which is not $\prec$-preceeded by any other untraversed arc, otherwise choose the red arc. Traverse the chosen 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. Moreover, C is $\prec$-respecting by construction

since $L \subseteq M_{\prec}$. Hence, if we can show that C traverses all arcs from R then this implies $L = L_C$ and the proof is complete.

To this end, define for each node $v \in V$, the value $\mathrm{dist}(v,o)$ to 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 $\mathrm{dist}(v,o)$ that all arcs emanating from v are contained in C.

If $\mathrm{dist}(v,o)=0$ 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 $t \geqslant 0$ and let $v \in V$ with $\mathrm{dist}(v,o)=t+1$. Let $a=(v,w)$ be the unique red arc emanating from v. Then $\mathrm{dist}(w,o)=t$ 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. □

Corollary 11. *Let $H=(V,R)$ be a graph, $o \in V$ and $\prec$ a source-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. Suppose that H is $\prec$-Eulerian with start o, and let C be an $\prec$-respecting Eulerian cycle with start o in H. Then $L_C \subseteq M_{\prec}$. Thus Statement 1 implies 2. The other direction is an immediate consequence of Theorem 10. □

The above corollary 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 possible set of last arcs. By the remark from above this check can be essentially performed by testing whether $M_{\prec}$ contains a directed spanning tree D rooted towards o (which can be done in linear time).

5. A polynomial time algorithm for S-DARP on paths

We now present Algorithm Alg-Path which solves S-DARP on paths. Let $G=(V,E,A)$ be a mixed graph such that $G[E]$ is a path. We assume throughout this section that Assumption 5 holds.

The algorithm starts in Step 2 by determining a suitable balancing set $B \sqsubset \vec{E}$ which is guaranteed to be contained in some optimal solution (following the results of Lemma 8). At this point, the graph $G[A+B]$ is degree balanced but may consist of several connected components. In order to turn the graph $\prec$-Eulerian with start o, the idea is to connect the components by pairs of antiparallel arcs from set $\vec{E}$ and in the same moment ensuring the existence of a possible set of last arcs.

This task is performed in two parts by the algorithm: In Step 3, a directed spanning tree rooted towards o of minimum cost is computed with respect to an auxiliary cost function. This cost function is adjusted to measure only the additional arcs that are not yet contained in $A+B$. In Step 5 an auxiliary arc set N is defined which contains those additional arcs together with their inverse arcs. This guarantees that $G[A+B+N]$ is in fact $\prec$-Eulerian which is proved formally in the following lemma:

Lemma 12. *The set $B+N$ returned by Algorithm* Alg-Path *is a feasible solution for* S-DARP, *i.e.*, *$G[A+B+N]$ is $\prec$-Eulerian with start o.*

Proof. Since $G[A+B]$ is degree balanced and N consists of pairs of anti-parallel arcs, also $G[A+B+N]$ is degree balanced. By construction, $G[A+B+N]$ contains a directed spanning tree rooted towards o, namely the tree D computed in Step 3. Hence it is strongly connected and Eulerian.

The set L of arcs determined in Step 4 is clearly a possible set of last arcs. The claim now follows from Theorem 10. □

It remains to show that the solution produced by Algorithm Alg-Path is not only feasible but also of minimum cost.

Theorem 13. *Algorithm* Alg-Path *finds an optimal solution for* S-DARP *on paths.*

Proof. Let S^* be an optimal solution such that $B \subseteq S^*$ (by Lemma 8 such a multiset S^* exists). By feasibility of S^* the graph $G[A+S^*]$ is $\prec$-Eulerian with start o.

Consider the multi-set $Z := (A+S^*)-(A+B) = S^*-B$. Since $G[A+B]$ and $G[A+S^*] = G[A+B+Z]$ are degree balanced and $Z \cap (A+B) = \emptyset$, we can decompose the set Z into arc disjoint cycles. Since Z consists of (multiple copies of) arcs from $\vec{E}$ 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 $G[A+S^*]$ and let L be its last set of arcs. Notice that $L \subseteq B + M_\prec + Z$, where $M_\prec$ is defined in Step 1 of the algorithm. The set L must contain a directed spanning tree D' rooted towards o. We partition D' into the sets $D'_{B+M_\prec} := D' \cap (B+M_\prec)$ and $D'_Z := D' \cap Z$. Thus, $c'(D'_{B+M_\prec}) = 0$ 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

$$c(Z) \geqslant 2c(D'_Z) = 2c'(D'_Z) + 2c'(D'_{B+M_\prec}) = 2c'(D') \geqslant 2c'(D). \tag{2}$$

Here, D is the directed spanning tree of minimum weight computed in Step 3. The set N computed in Step 5 has cost

$$c(N) = 2c(D \setminus (B+M_\prec)) = 2c'(D \setminus (B+M_\prec)) = 2c'(D) \overset{(2)}{\leqslant} c(Z). \tag{3}$$

Using this result yields that

$$\begin{aligned} c(A+B+N) &= c(A+B) + c(N) = c(A + (S^* \setminus Z)) + c(N) \\ &\overset{(3)}{\leqslant} c(A + (S^* \setminus Z)) + c(Z) = c(A+S^*). \end{aligned}$$

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

We briefly comment on the running time of Algorithm Alg-Path. A balancing set B can be found in time $\mathcal{O}(nm_A)$ by techniques as shown in [5]. 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 $\vec{E}$. 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 [17]. Thus Algorithm Alg-Path can be implemented to run in time $\mathcal{O}(n+m_A+\min\{(m_A+n)\log n, n^2\})$.

6. An approximation algorithm for general graphs

In this section we present our approximation algorithm for S-DARP on general graphs. The algorithm uses ideas similar to the ones in [12]. In this section we will assume tacitly that Assumption 6 is satisfied.

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 TSP-tour

Algorithm 1: Algorithm Alg-Path for S-DARP on paths.

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 source-order $\prec$

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

2 Compute a balancing set $B \sqsubset \vec{E}$ such that $B \subseteq S^*$ for some optimal solution S^*.

3 Compute a directed spanning tree D rooted towards o in $G[B+M_\prec+\vec{E}]$ of minimum weight $c'(D)$, where cost function c' is defined as follows:

$$c'(r)=\begin{cases} 0 & \text{if } r \in B+M_\prec, \\ c(r) & \text{if } r \in \vec{E} \setminus (B+M_\prec). \end{cases}$$

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

5 Let $D_+{:=}D-(B+M_\prec)$ and $N{:=}\vec{E} \cap (D_+ \cup D_+^{-1})$.
{ The set N contains the set D_+ of "new arcs" from the tree D and their inverses D_+^{-1}}.

6 Use the method from Theorem 10 to find a $\prec$-respecting Eulerian cycle C with start o in $G[A+B+N]$ such that L is the set of last arcs of C.

7 **return** the multiset $B+N$ and the cycle C.

that visits each vertex v with $d_A^+(v)>0$. Then, it uses this TSP-tour to obtain a feasible solution for S-DARP: The solution traverses along the TSP-tour, and whenever a vertex v is reached where arcs are emanating from, for each arc in A_v the TSP-tour is augmented

by a loop traversing that arc and a shortest path back to vertex v. The algorithm is displayed in Algorithm 2.

We now prove a bound on the quality of the solution found by the TSP-based algorithm Alg-TSP.

Lemma 14. *If in Step* 3 *of* Alg-TSP *a* ρ_{TSP}*-approximation algorithm for computing a TSP-tour is employed, then the algorithm finds a solution of cost at most* $\rho_{TSP}\,OPT + 2c(A)$.

Proof. Let S^* be an optimum augmenting set and C^* be a $\prec$-respecting Eulerian cycle in $G[A+S^*]$ starting at o. Since C^* visits all vertices from V_{source} (the set of vertices from V which are sources of arcs from A) the length of C^* (which equals OPT) is at least that of a shortest TSP-tour on V_{source}. Thus, the tour computed in Step 3 will have length at most $\rho_{TSP}\,\text{OPT}$. The additional cost incurred in Step 7 is not greater than $2c(A)$, since each path added has

Algorithm 2: TSP-based approximation algorithm Alg-TSP for S-DARP.

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

1 Let V_{source} be the set of vertices which are sources of arcs from A.
2 Compute a complete undirected auxiliary graph U with vertex set V_{source}. 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 TSP tour p in U. Assume that p visits the nodes of V_{source} in order $(o=v_0,v_1,\ldots,v_s,v_{s+1}=o)$.
4 Construct a feasible tour C for S-DARP as follows:
5 Start with the empty tour C.
6 **for** i:=0,…,s **do**
7 Assume that $A_{v_i}=\{a_1,\ldots,a_k\}$ with $a_i \prec a_j \Rightarrow i<j$.
8 For $j=1,\ldots,k$, let p_j be a directed shortest path in $G[\vec{E}]$ from the endpoint of a_j to v_i.
9 Add the k loops $a_1\,p_1,\ldots,a_k\,p_k$ to C.
10 Append to C the directed shortest path in $G[\vec{E}]$ from v_i to v_{i+1}.
11 **end for**
12 Let $S \leftarrow C-A$.
13 **return** the set S and the cycle C.

the weight of the corresponding arc from A. □

Since the cost of the optimum tour serving all jobs is at least $c(A)$, Lemma 14 implies that Alg-TSP is a $(\rho_{TSP}+2)$-approximation algorithm for S-DARP. Using Christofides' algorithm [7] we get $\rho_{TSP}=3/2$ and thus Alg-TSP implies a 7/2-approximation for S-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 5 for paths. We first compute a set of balancing arcs B which makes $G[A+B]$ degree balanced. Again, we then compute a rooted tree directed towards the origin o of minimum cost, double the new arcs which are not yet in $A+B$ and add the resulting set N to the solution. Alg-Last-Arcs is shown in Algorithm 3.

By a proof similar to Lemma 12 it follows that the set $B+N$ found by Algorithm Alg-Last-Arcs is indeed a feasible solution.

Lemma 15. *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.*

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

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

1 Compute a balancing multiset $B \sqsubset \vec{E}$ of minimum cost.
{**How this step can be accomplished with the help of a minimum cost flow** computation is described in detail in Lemma 15}**.**

2 Follow steps 1 and 3 to 6 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 $B+N$ and the cycle C

Proof. Let $S^* \sqsubset \vec{E}$ be an optimal solution, i.e., an augmenting multiset of arcs from $\vec{E}$ with minimum cost. Then the graph $G[A+S^*]$ is $\prec$-Eulerian with start o. Thus, the addition of the arcs from S^* turns G Eulerian, in particular degree balanced. Thus, the cost $c(S^*) = \mathrm{OPT} - c(A)$ is at least that of a minimum cost set $B \sqsubset \vec{E}$ which achieves the degree balance.

Step 1 can be carried out by performing a minimum cost flow computation in the auxiliary graph $F=(V,\vec{E})$. A vertex v has charge $d_G^-(v) - d_G^+(v)$ and the cost of sending one unit of flow over arc $r \in \vec{E}$ 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 multiset B. □

We continue to prove an upper bound on the cost of the set N of new arcs and their inverses computed in Step 2 of Alg-Last-Arcs.

Lemma 16. *The cost of the arc set N computed by* Alg-Last-Arcs *is at most $2(OPT - c(A))$.*

Proof. The proof of the lemma is similar to the one for Theorem 13. The major difference is that in general we cannot 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 $G[A+S^*]$. 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 $L-(A+B)=L-(M_\prec+B)$. Similar to Theorem 13 we can now conclude that

$$\mathrm{OPT}-c(A)=c(S^*)\geqslant c(L-(A+B))=c(L-(M_\prec+B))=c'(L)\geqslant c(N)/2.$$

This shows the claim. □

Lemmas 15 and 16 imply the following bound on the performance of Algorithm Alg-Last-Arcs:

Corollary 17. Alg-Last-Arcs *finds a solution of cost at most* $3OPT-2c(A)$.

Proof. By Lemma 15, $c(A+B)\leqslant\mathrm{OPT}$. Lemma 16 establishes that $c(N)\leqslant 2\mathrm{OPT}-2c(A)$. Thus $c(A+B+N)\leqslant 3\mathrm{OPT}-2c(A)$ as claimed. □

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

Theorem 18. *Algorithm* Alg-Combine *has a performance of* $\frac{1}{2}(\rho_{TSP}+3)$.

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

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

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

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

This shows the claim of the theorem. □

Using Christofides' algorithm [7] with $\rho_{TSP}=3/2$ results in a performance guarantee of $3/4+3/2=9/4$ for algorithm Alg-Combine.

Corollary 19. *There is an approximation algorithm for* S-Darp *with performance* $9/4$. *This algorithm can be implemented to run in time* $\mathcal{O}(\max\{n^3+m_Am_E+m_An\log n, m_E^2\log n+m_En\log^2 n\})$.

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 $\mathcal{O}(m_Am_E+m_An\log n)$. 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 [1]. □

7. Improved approximation algorithm 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 19. In particular, for trees where the TSP can be solved in polynomial time Theorem 18 already implies a 2-approximation algorithm. However, we can still improve this performance guarantee.

Theorem 20. *There exists a polynomial time approximation algorithm for* S-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 $B \sqsubset S^*$ as in Lemma 8. After the balancing we remove all vertices which are not incident to the arcs in $A + B$ 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 $I' = (G = (V, E, A + B), c, o, \prec)$ of S-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 8), we get that $\mathrm{OPT}(I) = \mathrm{OPT}(I')$.

Now look at the instance I'' of S-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 of I'' to a feasible solution of I' (by replacing arcs not in $\vec{E}$ 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 $\mathrm{OPT}(I) = c(A + B) + c(Z) = \mathrm{OPT}(I'') = c(A + B) + c(Z'')$, we have that $c(Z) = c(Z'')$.

Let $A + B + N$ be the solution of instance I found by the modified version of Alg-Last-Arcs. Then, using the arguments of Lemma 16 we get that

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

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 the modified Alg-Last-Arcs and Alg-TSP

by the techniques from the proof of Theorem 18: if $c \geqslant \frac{1}{3}\mathrm{OPT}$, then the cost of the solution produced by the first algorithm is bounded by

$$2\mathrm{OPT} - c(A) \leqslant 2\mathrm{OPT} - \tfrac{1}{3}\,\mathrm{OPT} = \tfrac{5}{3}\,\mathrm{OPT}.$$

Otherwise, for $c \leqslant \frac{1}{3}\,\mathrm{OPT}$, the cost of the solution produced by the second algorithm is bounded by

$$\mathrm{OPT} + 2c(A) \leqslant \mathrm{OPT} + \tfrac{2}{3}\,\mathrm{OPT} = \tfrac{5}{3}\,\mathrm{OPT}.$$

This yields an overall 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 [15,16]). 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 $\mathcal{O}(nm_E + n^2 \log n) = \mathcal{O}(n^2 \log n)$ [8,1]. All other steps can be carried out in time $\mathcal{O}(n^2)$ where again the algorithm from [17] is employed for computing a minimum weight directed spanning tree. □

8. Hardness results

Since S-DARP generalizes DARP, it follows from the hardness result in [11] that S-DARP is NP-hard even on trees. We show that this hardness continues to hold even if the source-order $\prec$ is a total source-order. We can also strengthen the hardness result of [11] and show that DARP is hard on caterpillar graphs. This is in contrast to an application of DARP in the next section where caterpillar graphs naturally arise.

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*. We restrict the class of caterpillars further to those graphs where no two hairs are incident, i.e., the nodes on the backbone are of maximum degree 3.

Theorem 21. DARP *and* S-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* S-DARP *remain true if the source-ordering is restricted to be total.*

Proof. We first address the hardness of DARP. The hardness is shown by a reduction from the Steiner tree problem on bipartite graphs, BIPARTITE-STP. An instance of BIPARTITE-STP consists of a bipartite graph $H = (X \cup Y, F)$ and a nonnegative number

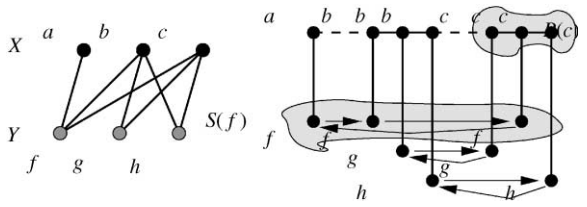

Fig. 2. Transformation of the Steiner tree problem on a bipartite graph into DARP on a caterpillar in Theorem 21.

$k \leqslant |F|$. It is NP-complete to decide whether there exists a subtree of H that spans all the vertices in Y and has at most k edges [13, Problem ND12].

One can make two assumptions on H without loss of generality: First, each vertex in Y has degree at least two. Otherwise, for a vertex $y \in Y$ with degree one, there is no choice than including the unique edge incident on y in the Steiner tree. Second, H is connected. Otherwise, either there is a connected component containing Y, or H cannot contain a Steiner tree for the set Y.

Let $H = (X \cup Y, F)$ be an instance of BIPARTITE-STP. We create an instance $I = (G = (V, E, A), c, o)$ of DARP. The construction of graph G is illustrated in Fig. 2. Start with a graph consisting of $2|F|$ nodes and $|F|$ pairwise nonincident edges. For each edge $[x, y] \in F$ where $x \in X$ and $y \in Y$, choose a yet unlabeled edge in G and label its endpoints by x and y, respectively. Now create the backbone of the caterpillar graph by inserting a path of $|F| - 1$ additional edges. These backbone edges are inserted between nodes with labels from X in such a way that for each $x \in X$ the graph induced by the nodes labeled x in G is a connected path, denoted by $P(x)$.

Set the weight of a backbone edge with two endpoints sharing the same label to 0, and the weight of backbone edges with two different labels to some large number $M := 2|F| + 1$. Set the weight of a hair to 1.

The arc set A is constructed as follows: For $y \in Y$ denote by $S(y)$ the set of foot vertices in G labeled with y. Then, for each $y \in Y$, choose a directed simple cycle connecting $S(y)$ and add its arc set to A. Finally, the origin o of the server is chosen to be the source of an arbitrary arc in A. Observe that by construction the graph $G[A]$ is degree balanced. It consists of the set of connected components $\{S(y) \mid y \in Y\}$. Each of the components is strongly connected and Eulerian.

Let $C = \sum_{a \in A} c(a)$. We claim 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 $C + 2k$.

Suppose that T is a Steiner tree in H with at most k edges connecting the vertices in Y. We construct a multiset S of arcs, $S \sqsubset \vec{E}$, such that $G[A + S]$ is Eulerian and contains o and $c(A + S) \leqslant C + 2k$: For each $x \in X$ spanned by T, add all arcs from the set $\vec{P}(x)$ to S (these arcs are of cost 0). For each edge $[x, y] \in T$, add a pair of antiparallel arcs between the endpoints of the unique hair labeled $[x, y]$ to multiset S (these arcs are of cost 1 each). By this construction, graph $G[A + S]$ is degree

balanced. Since T was spanning Y and connected, $G[A+S]$ is strongly connected and hence Eulerian. Further, $c(A+S) \leqslant C+2k$. This shows the first direction.

Assume conversely that S, $S \sqsubset \vec{E}$, is a feasible solution for instance I, and its cost satisfy $c(A+S) \leqslant C+2k$. Then $G[A+S]$ is Eulerian and contains o. The set S cannot contain any arc of cost M, since otherwise $c(A+S) = c(A) + c(S) \geqslant C + M = C + 2|F| + 1 > C + 2k$.

We now define a subgraph T of H as follows: For each $x \in X$, $y \in Y$, if S contains (at least one copy) of an arc between a vertex in $P(x)$ and $S(y)$ in arbitrary direction, then the edge $[x, y]$ is included in T. Since $G[A]$ and $G[A+S]$ are degree balanced and $S \cap A = \emptyset$, we can decompose S into arc disjoint cycles. Since $S \sqsubset \vec{E}$ 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. Hence, T consists of 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_1 and y_2 be two arbitrary vertices from Y. Since y_1, y_2 are incident with arcs from A, and $G[A+S]$ is strongly connected, there is a directed path $(r_1, \ldots, r_t)$ from a node labeled y_1 to one labeled y_2 in $G[A+S]$. The node labels along this path change only when r_i is of type $r_i = (x, y)$ or $r_i = (y, x)$ for suitable $x \in X$ and $y \in Y$. By construction, tree T contains edge $[x', y']$ in this case. Hence, T connects y_1 and y_2. This completes the proof of the hardness results for DARP.

Since $|A_v| \leqslant 1$ for all nodes v in the construction used above, by choosing $\prec$ to be the empty relation the hardness for S-DARP immediately follows. □

9. S-DARP with start and stop penalties

In this section we show how to extend our results to the case when there are start- and stop-penalties for the service vehicle, which makes the problem more realistic in view of applications. In elevator systems the time that the elevator needs to accelerate or decelerate in order to pick up or deliver its load can usually not be neglected. Thus, it is natural to penalize each stop and start of the server on its route.

In the Dial-a-Ride-Problem with penalties, short PENALTY-S-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-S-DARP in a meaningful way as a graph augmentation problem, we have to allow augmenting arcs from $V \times V$ and not just from $\vec{E}$, since each arc corresponds to a move and incurs a start and stop penalty. The cost function $c : E \to \mathbb{R}_{\geqslant 0}$ 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 22 (*Graph augmentation version of Penalty*-S-DARP)**.** An instance of PENALTY-S-DARP consists of the same input as for S-DARP together with additional

penalty functions $p^+, p^- : V \to \mathbb{R}_{\geqslant 0}$ on the set of vertices V. The objective is to find a multiset S of arcs, $S \sqsubset V \times V$, minimizing the weight

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

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

In the sequel we show that an instance $I = (G = (V,E,A), c, o, \prec, p^-, p^+)$ of Penalty-S-Darp can be transformed into an equivalent instance of S-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 $c'(e_v) = (p^+(v) + p^-(v))/2$. 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' is induced naturally by that on A. Finally, the start vertex o' equals o.

Lemma 23. *Let $I = (G, c, o, \prec, p^+, p^-)$ be an instance of* Penalty-S-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 instance I' of S-Darp where S' is an augmenting set of arcs. Let C' be a $\prec$-respecting Eulerian cycle in $G'[A' + S']$ 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 for each arc (u,v) in S the arc $(u^{(\pm)}, v^{(\pm)})$ to S'.

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 S-Darp on trees can be applied directly to Penalty-S-Darp on trees. Similarly, approximation results for general graphs carry over immediately. Hence, we obtain the following result:

Corollary 24. *The problem* Penalty-S-Darp *can be approximated on trees with performance* 5/3 *and with performance* 9/4 *on general graphs.*

Table 1
Complexity and approximation results for DARP and related problems

Graph class	S-DARP	DARP	PENALTY-S-DARP
Paths	Polynomial time solvable (Theorem 13)	Polynomial time solvable [5]	NP-hard (Theorem 25)
			Approximable within 5/3. (Corollary 24)
Trees	NP-hard, even on caterpillars (Theorem 21)	NP-hard, even on caterpillars (Theorem 21)	NP-hard
	Approximable within 5/3 (Theorem 20)	Approximable within 5/4 [11]	Approximable within 5/3. (Corollary 24)
General Graphs	NP-hard Approximable within 9/4 (Corollary 19)	NP-hard [12] Approximable within 9/5 [12]	NP-hard Approximable within 9/4 (Corollary 24)

However, transforming an instance of PENALTY-S-DARP where $G[E]$ is a path yields an instance of S-DARP where $G'[E']$ is a caterpillar graph. This seems unfortunate, since we know from Theorem 21 that S-DARP is NP-hard to solve on caterpillars. Is there a better transformation? More general, is PENALTY-S-DARP on paths still polynomial time solvable?

The caterpillar constructed in the proof of Theorem 21 has the property that jobs have sources and targets only in the feet of the caterpillar. Actually every instance of S-DARP on caterpillars with these properties can be transformed into an equivalent instance of PENALTY-S-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 23 that the constructed instance of PENALTY-S-DARP on the path (which corresponds to the former backbone) is in fact an equivalent instance to the instance of S-DARP on the caterpillar. Thus, we obtain the following result which contrasts with the polynomial solvability of S-DARP on paths:

Theorem 25. PENALTY-S-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 [2]. We have shown that even in the presence of source-order constraints for the transportation jobs the problem can be solved in polynomial time on paths which generalizes the result of [5]. On trees, however, the problem is NP-hard.

Table 1 gives an overview on the results for S-DARP obtained in this paper and the known results from literature for DARP. The last column addresses the problem PENALTY-S-DARP which is the extension of S-DARP with start- and stop-penalties discussed in Section 9.

References

[1] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, Networks Flows, Prentice-Hall, Englewood Cliffs, NJ, 1993.
[2] N. Ascheuer, M. Grötschel, S.O. Krumke, J. Rambau, Combinatorial online optimization, Proceedings of the International Conference of Operations Research (OR'98), Springer, Berlin, 1998, pp. 21–37.
[3] N. Ascheuer, S.O. Krumke, J. Rambau, Competitive scheduling of elevators, preprint SC 98-34, Konrad-Zuse-Zentrum für Informationstechnik Berlin, November 1998.
[4] N. Ascheuer, S.O. Krumke, J. Rambau, Online dial-a-ride problems: minimizing the completion time, Proceedings of the 17th International Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science, Vol. 1770, Springer, Berlin, 2000, pp. 639–650.
[5] M.J. Atallah, S.R. Kosaraju, Efficient solutions to some transportation problems with applications to minimizing robot arm travel, SIAM J. Comput. 17 (5) (1988) 849–869.
[6] M. Charikar, B. Raghavachari, The finite capacity dial-a-ride problem, Proceedings of the 39th Annual IEEE Symposium on the Foundations of Computer Science, 1998.
[7] N. Christofides, Worst-case analysis of a new heuristic for the traveling salesman problem, Technical Report, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, 1976.
[8] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, MIT Press, Cambridge, MA, 1990.
[9] M. Dror, H. Stern, P. Trudeau, Postman tour on a graph with precedence relation on the arcs, Networks 17 (1987) 283–294.
[10] E. Feuerstein, L. Stougie, On-line single server dial-a-ride problems, Theoret. Comput. Sci. (2001), to appear.
[11] G.N. Frederickson, D.J. Guan, Nonpreemptive ensemble motion planning on a tree, J. Algorithms 15 (1) (1993) 29–60.
[12] G.N. Frederickson, M.S. Hecht, C.E. Kim, Approximation algorithms for some routing problems, SIAM J. Comput. 7 (2) (1978) 178–193.
[13] M.R. Garey, D.S. Johnson, Computers and Intractability (A guide to the theory of NP-completeness), W.H. Freeman and Company, New York, 1979.
[14] D.J. Guan, Routing a vehicle of capacity greater than one, Discrete Appl. Math. 81 (1) (1998) 41–57.
[15] D. Harel, R.E. Tarjan, Fast algorithms for finding nearest common ancestors, SIAM J. Comput. 13 (2) (1984) 338–355.
[16] B. Schieber, U. Vishkin, On finding lowest common ancestors: simplification and parallelization, SIAM J. Comput. 17 (6) (1988) 1253–1262.
[17] R.E. Tarjan, Finding optimum branchings, Networks 7 (1977) 25–35.