Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

SVEN O. KRUMKE JÖRG RAMBAU STEFFEN WEIDER

An Approximation Algorithm for the Non-Preemptive Capacitated Dial-a-Ride Problem

ZIB-Report 00-53 (December 2000)

AN APPROXIMATION ALGORITHM FOR THE NON-PREEMPTIVE CAPACITATED DIAL-A-RIDE PROBLEM

SVEN O. KRUMKE [1], JÖRG RAMBAU [1], AND STEFFEN WEIDER [1]

ABSTRACT. In the Capacitated Dial-a-Ride Problem (CDARP) we are given a 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 C objects at a time can move on the transportation network in order to process the transportation requests. The problem CDARP consists of finding a shortest transportation for the jobs starting and ending at a designated start location.

In this paper we are concerned with the restriction of CDARP to graphs which are simple paths. This setting arises for instance when modelling applications in elevator transportation systems. It is known that even for this restricted class of graphs CDARP is NP-hard to solve. We provide a polynomial time approximation algorithm that finds a transportion of length at most thrice the length of the optimal transportation.

1. INTRODUCTION

In the *Capacitated Dial-a-Ride Problem* CDARP we are given a transportation network and a finite set of transportation jobs (requests). Each request specifies the source and target location which are both part of the network. A server of capacity C, that is, a server which is able carry at most C objects at a time, can move on the transportation network in order to process the transportation requests. The problem CDARP consists of finding a shortest transportation schedule for the server starting and ending at a designated start location and serving all specified requests.

In this paper we are concerned with the restriction of CDARP to graphs which are simple paths. This setting arises for instance when modelling the transporation system consisting of a single elevator.

This paper is organized as follows. In Section 2 we formally define the problem CDARP. We also give a short overview over previous work on the problem. The main contribution of this paper is contained in Sections 3 and Section 4. In Section 3 we present a polynomial time approximation algorithm for CDARP on paths with performance 3. The proof is contained in Section 4. Section 5 illustrates our algorithm on a sample instance.

It should be noted that in [CR98] the authors claimed an approximation algorithm with performance 2 for CDARP on paths but neither the algorithm nor a proof of its performance was given.

---

[1]Konrad-Zuse-Zentrum für Informationstechnik Berlin, Department Optimization, Takustr. 7, D-14195 Berlin-Dahlem, Germany. Email: {krumke,rambau,weider}@zib.de. Research supported by the German Science Foundation (DFG, grant Gr 883/5-3)

Key words and phrases. NP-completeness, polynomial-time approximation algorithms, stacker-crane problem, vehicle routing, elevator system.

2. Preliminaries and Problem Definition

2.1. **Basic Notation.** 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 $X(u) \geq 1$. 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

$$\begin{aligned}(X+Y)(u) &= X(u) + Y(u)\\ (X-Y)(u) &= \max\{X(u) - Y(u), 0\}\\ (X \cap Y)(u) &= \min\{X(u), Y(u)\}.\end{aligned}$$

The multiset $X \sqsubset U$ is a subset of the multiset $Y \sqsubset U$, denoted by $X \subseteq Y$, if $X(u) \leq Y(u)$ 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* $G = (V, E, R)$ consists of a set V of vertices, a set E of undirected edges without parallels, and a multiset R of directed arcs (parallel arcs allowed). An edge with endpoints u and v will be denoted by $[u, v]$, an arc r from u to v by (u, v). In the latter case we write $u = \alpha(r)$ and $v = \omega(r)$.

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. The *out-degree* of a vertex v in G, denoted by $\deg^+_G(v)$, equals the number of arcs in G leaving v. Similarly, the *in-degree* $\deg^-_G(v)$ is defined to be the number of arcs entering v. If $X \sqsubset A$, we briefly write $\deg^+_X(v)$ and $\deg^-_X(v)$ instead of $\deg^+_{G[X]}(v)$ and $\deg^-_{G[X]}(v)$.

2.2. **Problem Definition.** An instance of CDARP is given by a finite mixed graph $G = (V, E, R)$ with edge weights given by $d: E \to \mathbb{R}_{\geq 0}$, a capacity $C \in \mathbb{N}$ and start position $o \in V$ for the server. The "undirected part" $G[E]$ of G models a transportation network, the set of arcs R represents requests or objects to be transported. Request $r \in R$ must be moved by the server from vertex $\alpha(r) \in V$ (the source of request r) to vertex $\omega(r) \in V$ (the destination of r).

In this paper we are concerned with the situation that $G[E]$ is a simple path and o is one of its end points. For notational convencience we assume that $V = \{1, \dots, n\}$, $E = \{ [i, i+1], i = 1, \dots, n-1 \}$ and that the initial position o of the server is 1.

A *move* of the server from vertex v to vertex w carrying a set Q of requests is denoted by the triple (v, w, Q). It is required that for any move the number $|Q|$ of requests carried by the server does not exceed its capacity C. A move with $Q = \emptyset$ is called *empty move*, a move with $|Q| = C$ is called a *fully loaded move*. A *transportation* is a sequence of moves of the form $T = (v_1, v_2, Q_1), (v_2, v_3, Q_2), \dots, (v_k, v_{k+1}, Q_k)$. The *cost* of such a transportation T is $\sum_{i=1}^{k} d(v_i, v_{i+1})$, where $d(v_i, v_{i+1})$ denotes the shortest path distance between vertices v_i and v_{i+1} in G. Request r is moved from vertex v_1 to vertex v_{k+1} by T if $r \in Q_i$ for $i = 1, \dots, k$.

A *feasible non-preemptive transportation* for an instance of CDARP is a transportation $T = (v_1, v_2, Q_1), (v_2, v_3, Q_2), \dots, (v_k, v_{k+1}, Q_k)$ which satisfies the following conditions:

(i) T starts and ends at o, that is, $v_1 = v_{k+1} = o$.
(ii) For any request $r \in R$ the subsequence consisting of those moves (v_i, v_{i+1}, Q_i) from T with $r \in Q_i$ is a contiguous subsequence of T and forms a transportation from $\alpha(r)$ to $\omega(r)$.

In the case of a preemptive transportation the requirement in (ii) that the subsequence is a contiguous subsequence of T is dropped.

Definition 2.1 (Problem CDARP)**.** An instance of CDARP is given by a finite mixed graph $G = (V, E, R)$ with edge weights given by $d\colon E \to \mathbb{R}_{\geq 0}$, a capacity $C \in \mathbb{N}$ and start position $o \in V$ for the server. The goal of problem CDARP is to find a feasible transportation with minimum cost.

The optimum cost of a feasible transportation for instance I of CDARP is denoted by $\mathsf{OPT}(I)$. We drop the reference to I provided no confusion can occur. The problem CDARP is NP-hard even on paths even if the capacity of the server is two [Gua98]. Hence, we are interested in efficient *approximation algorithms* for CDARP.

Definition 2.2 (Approximation Algorithm)**.** An *approximation algorithm with performance guarantee* γ for CDARP is a polynomial time algorithm ALG which, given any instance I of CDARP, finds a feasible transportation with cost $\mathsf{ALG}(I)$ such that

$$\mathsf{ALG}(I) \leq \gamma\, \mathsf{OPT}(I).$$

In all what follows we assume without loss of generality that for a given instance of CDARP at least one arc of R is incident with vertex n. Notice that this implies $\mathsf{OPT} \geq 2d(1, n)$, since any feasible transportation must visit vertex n and return to its starting position at $o = 1$. We also assume that the number of requests $|R|$ is at least C. If this is not the case, then a single upward and downward motion of the server from 1 to n and backwards can be used to get a transportation of cost $2d(1, n)$ which by the previous comment must then be optimal.

2.3. **Previous Work.** As noted before Guan showed that CDARP is NP-hard even on paths even if the capacity of the server is two [Gua98]. The preemptive version is polynomial time solvable on paths [Gua98], but NP-hard even on trees and even if the capacity of the server is one [FG93]. In [CR98] an approximation algorithm for CDARP with performance $\mathcal{O}(\sqrt{C} \log n \log\log n)$ was given, where C denotes the capacity of the server. In the same paper the authors claimed an approximation algorithm with performance 2 for CDARP on paths but neither the algorithm nor a proof of its performance was given.

The problem CDARP with capacity $C = 1$ is also called DARP or the *Stacker-Crane-Problem*. In [FHK78] the authors present a $9/5$-approximation algorithm for DARP 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].

3. The Algorithm

In our presentation we imagine the simple path $G[E]$ as a vertical line with vertex 1 being the lowest and vertex n the highest vertex, see Figure 1 for an illustration.

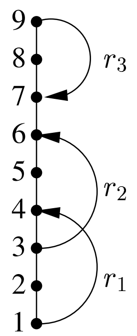

FIGURE 1. Example of a mixed graph $G = (V, E, R)$ given in an instance of CDARP.

We define the set of *upward requests* and *downward requests* as follows:

$$R^{\uparrow} = \{ r \in R : \alpha(r) < \omega(r) \}$$
$$R^{\downarrow} = \{ r \in R : \alpha(r) > \omega(r) \}.$$

In the example presented in Figure 1 we have $R^{\uparrow} = \{r_1, r_2\}$ and $R^{\downarrow} = \{r_3\}$. Let A be a subset of R which is either completely contained in $R^{\uparrow}$ or $R^{\downarrow}$. In this case we let

$$\alpha(A) = \begin{cases} \min\{ \alpha(r) : r \in A \} & \text{, if } A \subseteq R^{\uparrow} \\ \max\{ \alpha(r) : r \in A \} & \text{, if } A \subseteq R^{\downarrow} \end{cases}$$

and

$$\omega(A) = \begin{cases} \max\{ \omega(r) : r \in A \} & \text{, if } A \subseteq R^{\uparrow} \\ \min\{ \omega(r) : r \in A \} & \text{, if } A \subseteq R^{\downarrow}. \end{cases}$$

In the sequel we have to refer to those request arcs r which have to be transported over a specific edge $[v, v+1]$. To facilitate the presentation we define the notions of covers and segments.

Definition 3.1 (Cover)**.** Let $e = [v, v+1]$ be an edge in the graph $G[E]$. A request arc $r = (\alpha(r), \omega(r)) \in R$ is said to *cover* e if $\alpha(r) \leq v$ and $\omega(r) \geq v+1$ or $\alpha(r) \geq v+1$ and $\omega(r) \leq v$.

Definition 3.2 (Segment)**.** Let $R' \subseteq R$. A *segment* of R' is an inclusionwise maximal subset $S \subseteq R'$ with the property that the set of edges from G covered by S forms a connected subpath of G.

Observe that the segments of a set $R' \subseteq R$ form a partition of R'. We need one final notation before presenting our algorithm.

Definition 3.3 (Number of covering requests μ_v)**.** For a vertex $v \in V \setminus \{n\}$ and a subset $D \subseteq R$ we define $\mu_v(D)$ be the number of requests in D that cover the edge $[v, v+1]$. We also set $\mu_n(R') := 0$. We omit the set R' if it is clear from the context.

We are now ready to state our approximation algorithm. The main algorithm FINDANDPASTE is shown in Algorithm 2. This algorithm uses the subroutines FINDUPARCS presented in Algorithm 1 and FINDDOWNARCS displayed in Algorithm 3.

Algorithm 1 Algorithm FINDUPARCS

Input: A multiset of requests $S \sqsubset R^{\uparrow}$, two vertices $v_L \leq v_U$ from V
{The multiset S is modified by FINDUPARCS and the modified set is returned.}

1 Let H be the subpath of G formed by the vertices $v_L, v_L + 1, \ldots, v_U$.
2 **if** there exists edges $[v, v+1]$ in H with $v_L \leq v < v+1 \leq v_U$ which are not covered by any arc from S **then**
3 For each of these edges $[v, v+1]$ add a dummy arc $(v, v+1)$ to S.
4 **end if**
5 $M_1 := \emptyset$, $M_2 := \emptyset$
6 $l = 1$, $u = 2$ {We maintain the invariant that $\omega(M_l) \leq \omega(M_u)$. Here, u stands for "upper" and l for "lower".}
7 **while** $M_u = \emptyset$ or $\omega(M_u) < \omega(S)$ **do**
8 Find a path P in the directed (acyclic) graph (V, S) with

$$\omega(M_l) \leq \alpha(P) \leq \omega(M_u) < \omega(P) \qquad (1)$$

such that $\omega(P)$ is maximum among all those paths. (Here we set $\omega(\emptyset) := \alpha(S)$).
9 Set $M_l := M_l \cup P$ {Add the arcs from P to the "lower" set M_l.}
10 $S := S - P$ {Remove the arcs from P from the multiset S.}
11 Interchange the values of l and u.
12 **end while**
13 **return** M_1, M_2 and the modified multiset S.

Before we analyze the performance of FINDANDPASTE we first derive some useful properties of the subroutines FINDUPARCS and FINDDOWNARCS.

Lemma 3.4. *Suppose that Algorithm* FINDUPARCS *is called with a nonempty set $S \subseteq R^{\uparrow}$ and that P is a path which is found in Step 8 of* FINDUPARCS. *Then this path P satisfies:*

$$\mu_{\omega(P)-1}(S) > \mu_{\omega(P)}(S)$$

for the current set S when P is added to M_l in Step 9.

Proof. Suppose that the claim were not true for some path P. Let $v = \omega(P)$. Since $\mu_{v-1}(S) \leq \mu_v(S)$, it follows that for any arc ending in v there must be at least one arc from S starting in v. However, P ends in v and thus we could extend P by at least one arc from S which starts in v. This contradicts the property that P was chosen in such a way that $\omega(P) = v$ is maximum. □

It is not trivial that in each iteration of FINDUPARCS a path P with the desired property (1) exists.

Lemma 3.5. *Suppose that Algorithm* FINDUPARCS *is called with a nonempty set $S \subseteq R^{\uparrow}$. Then in any iteration of Step 8 there exists a path P with the required properties.*

Algorithm 2 Algorithm FINDANDPASTE

Input: An instance of CDARP where $G[E]$ is a path.

1 Compute the set of upward requests $R^{\uparrow}$ and downward requests $R^{\downarrow}$.
2 $A := \emptyset$ {Each arc in the multiset A corresponds to a transportation. In a final "paste"-step these transportations will be pasted together by considering the DARP instance (V, E, A).}
3 **while** $R^{\uparrow} \neq \emptyset$ **do**
4   Let S be a segment of $R^{\uparrow}$
5   Let $L := \alpha(S)$ and $U := \omega(S)$.
6   Call Algorithm FINDUPARCS(S) C times with $v_L := L$ and $v_L := U$ to obtain $2C$ sets of arcs M_1^j and M_2^j, $j = 1, \dots, C$. (Notice that S shrinks with each call to FINDUPARCS provided it is not yet empty but the values of v_L and v_U remain fixed.)
7   Set $X_i^{\uparrow} := \biguplus_{i=1}^{C} P_i^j$ for $i = 1, 2$.
8   Construct two sequences of upward moves, U_1 and U_2, where U_i $(i = 1, 2)$ transports all objects from $X_i^{\uparrow}$. U_i starts at $\alpha(X_i^{\uparrow})$ and ends at $\omega(X_i^{\uparrow})$.
9   $A := A + (\alpha(X_1^{\uparrow}), \omega(X_1^{\uparrow}))$ {Add elements to the multiset A.}
10   **if** $X_2^{\uparrow} \neq \emptyset$ **then**
11     $A := A + (\alpha(X_2^{\uparrow}), \omega(X_2^{\uparrow}))$
12   **end if**
13   $R^{\uparrow} := R^{\uparrow} \setminus (X_1^{\uparrow} \cup X_2^{\uparrow})$
14 **end while**
15 **while** $R^{\downarrow} \neq \emptyset$ **do**
16   In the same way as above, call Algorithm FINDDOWNARCS C times and construct two sequences D_1 and D_2 of downward moves from the $2C$ sets of downward arcs. Add directed arcs $(\alpha(D_i), \omega(D_i))$ of D_i $(i = 1, 2)$ to the multiset A. Remove the downward arcs form the $2C$ sets found by FINDDOWNARCS from $R^{\downarrow}$.
17 **end while**
18 Consider the instance Π of DARP with underlying graph $G[E]$, request set A and start vertex $o = 1$. Each arc in A corresponds to one sequence of (upward or downward) moves constructed above in steps 8 and 16.
19 Find an optimal solution T_{Π} for Π in polynomial time with the help of the algorithm from [AK88].
20 Chain the sequences of upwards and downwards moves found in steps 8 and 16 to a transportation by taking them in the order as the corresponding arcs appear in T_{Π} and connecting them by empty moves, if necessary.

Proof. We show the claim by induction on the number of iterations. In the first iteration, we are in the situation that $M_u = M_l = \emptyset$. Hence, $\omega(M_u) = \omega(M_l) = \alpha(S)$ and condition (1) reduces to

$$\alpha(S) = \alpha(P) < \omega(P). \tag{2}$$

Clearly, there must be a path starting at $\alpha(S)$, since S is nonempty. Any such path satisfies (2).

Assume now that we have reached in the ith iteration ($i \geq 2$) and for all previous iterations it was possible to find a path. Let $P^{(i-1)}$ be the path found in the previous iteration $i-1$. To shorten notation we set $w := \omega(P^{(i-1)})$. Let $S^{(i-1)}$, $S^{(i)}$ denote the set S at the beginning of iteration $(i-1)$ and i, respectively. Notice that for the set M_u at the beginning of iteration i we have $\omega(M_u) = w$. Thus (2) can be restated as

$$\omega(M_l) \leq \alpha(P) \leq w < \omega(P) \tag{3}$$

Notice that at the beginning of the first iteration we have have $\mu_v(S^{(1)}) \geq 1$ for all $\alpha(S) \leq v < \omega(S)$ by Step 3. Since in all previous iterations we have we have removed only such arcs from S that end in vertices $u \leq w$, it follows that

$$\mu_u(S^{(i-1)}) = \mu_w(S^{(i)}) \quad \text{for all } u \geq w. \tag{4}$$

By Lemma 3.4 we have $\mu_{w-1}(S^{(i-1)}) > \mu_w(S^{(i-1)})$. Together with (4) we obtain

$$\mu_{w-1}(S^{(i)}) \geq \mu_w(S^{(i)}) \geq 1. \tag{5}$$

Our first step is to construct a path P formed by arcs from $S^{(i)}$ such that

$$\alpha(P) \leq \omega(M_u) = w < \omega(P). \tag{6}$$

To this end, we distinguish two cases.
Case 1: $\mu_{w-1}(S^{(i)}) \geq \mu_w(S^{(i)}) = \mu_{w+1}(S^{(i)}) = \cdots = \mu_{v-1}(S^{(i)}) > \mu_v(S^{(i)})$ for some $v > w$.

In this case, there must be least one arc from $S^{(i)}$ ending in v. Observe that for any vertex u with $w \leq u < v$ as many arcs from $S^{(i)}$ end in u as arcs emanate from u. Hence, it follows that there is a path P formed by arcs of $S^{(i)}$ that starts at some vertex $x \leq w$ and ends in $v > w$, which means that P satisfies (6).
Case 2: $\mu_{w-1}(S^{(i)}) \geq \mu_w(S^{(i)}) = \mu_{w+1}(S^{(i)}) = \cdots = \mu_{v-1}(S^{(i)}) < \mu_v(S^{(i)})$ for some $v > w$.

Either there exists an arc $r \in S^{(i)}$ with $\omega(r) = v$ and $\alpha(r) < v$ or there exists $r \in S^{(i)}$ with $\omega(r) > v$ and $\alpha(r) < v$ (where we have used the fact that $1 \leq \mu_w(S^{(i)}) = \mu_{w+1}(S^{(i)}) = \cdots = \mu_{v-1}(S^{(i)})$). Again, since for any vertex u with $w \leq u < v$ as many arcs from $S^{(i)}$ end in u as arcs emanate from u, we can conclude that there exists a path P satisfying (6).

We now show that for any path P satisfying (6) far, we have in fact that $\omega(M_l) \leq \alpha(P)$ which proves the claim of the lemma. If $M_l = \emptyset$ then there is nothing to show. Hence assume that $M_l \neq \emptyset$. Suppose that $\omega(M_l) > \alpha(P)$. It follows that in one of the previous iterations $j < i$ the path P satisfied $\alpha(P) \geq \omega({M_l}^{(j)})$ for the then current version ${M_l}^{(j)}$ of M_l. But this means that P it met all the conditions required in Step 8. Thus, the path chosen in iteration j was not maximum with respect to its end vertex. This is a contradiction. □

Corollary 3.6. *Suppose that Algorithm* FINDUPARCS *is called with a nonempty set* $S \subseteq R^{\uparrow}$. *Algorithm* FINDUPARCS *terminates after at most* $n-1$ *iterations with arc sets* M_1 *and* M_2 *such that for each edge* $[v, v+1]$ *with* $\alpha(S) \leq v < \omega(S)$ *there exists at least one and at most two arcs in* $M_1 \cup M_2$ *covering* $[v, v+1]$. *Moreover, all dummy arcs added in Step 3 are contained in* $M_1 \cup M_2$.

Proof. The property that FINDUPARCS terminates after no more than $n-1$ iterations follows from Lemma 3.5 and the fact that in each iteration $\omega(M_u)$ increases strictly.

We now consider the covering property. By Lemma 3.5 the path found in the first iteration starts at $\alpha(S)$. It is easy to show by induction that after each iteration all edges $[v, v+1]$ with $v < \omega(M_u)$ are covered. Hence, at termination all edges are covered at least once.

The arcs in M_1 do not overlap (where we call r and r' overlapping if $\alpha(r) < \alpha(r') < \min\{\omega(r), \omega(r')\}$ or vice versa) and neither do those in M_2. Hence we can conclude that any edge is covered by at most two arcs, that is at most one from M_1 and at most one from M_2. If a dummy arc $(v, v+1)$ was added in Step 3 then by construction it is the only arc covering edge $[v, v+1]$. Since we have shown that each edge is covered by the arcs in $M_1 \cup M_2$ it follows that the dummy arc must be contained in $M_1 \cup M_2$. □

We close this section by commenting on Algorithm FINDDOWNARCS which is needed in Step 16 of the main Algorithm FINDANDPASTE. FINDDOWNARCS works on $R^{\downarrow}$ in the analogous way as FINDUPARCS processes $R^{\uparrow}$. Basically the only difference is that the sets M_1 and M_2 "grow downwards" from $\alpha(S)$ to $\omega(S)$ instead of "growing upwards" as in FINDUPARCS.

Algorithm 3 Algorithm FINDDOWNARCS

Input: A multiset of requests $S \sqsubset R^{\downarrow}$, two vertices $v_L \leq v_U$ from V

1 Let H be the subpath of G formed by the vertices $v_L, v_L+1, \ldots, v_U$.
2 **if** there exists edges $[v, v+1]$ in H with $v_L \leq v < v+1 \leq v_U$ which are not covered by any arc from S **then**
3 For each of these edges $[v, v+1]$ add a dummy arc $(v+1, v)$ to S.
4 **end if**
5 $M_1 := \emptyset$, $M_2 := \emptyset$
6 $l = 1, u = 2$ {We maintain the invariant that $\omega(M_l) \leq \omega(M_u)$. Here, u stands for "upper" and l for "lower".}
7 **while** $M_l = \emptyset$ or $\omega(M_l) > \omega(S)$ **do**
8 Find a path P in the directed (acyclic) graph (V, S) with

$$\omega(M_u) \geq \alpha(P) \geq \omega(M_l) > \omega(P) \tag{7}$$

such that $\omega(P)$ is minimum among all those paths. (Here we set $\omega(\emptyset) := \alpha(S)$).
9 Set $M_u := M_u \cup P$ {Add the arcs from P to the "upper" set M_u.}
10 $S := S - P$ {Remove the arcs from P from the multiset S.}
11 Interchange the values of l and u.
12 **end while**
13 **return** M_1, M_2 and the modified multiset S.

The following property of FINDDOWNPATH can be proven analogously to the corresponding result about FINDUPPATH:

Lemma 3.7. *Suppose that Algorithm* FINDDOWNARCS *is called with a nonempty set $S \subseteq R^{\downarrow}$. Algorithm* FINDDOWNARCS *terminates after at most $n-1$ iterations with arc sets M_1 and M_2 such that for each edge $[v, v+1]$ with $\omega(s) \leq v < \alpha(S)$ there exists at least one and at most two arcs in $M_1 \cup M_2$ covering $[v, v+1]$. Moreover, all dummy arcs added in Step 3 are contained in $M_1 \cup M_2$.* □

4. Proof of Performance

In this section we are going to establish the performance of our algorithm, that is, the property that given any instance of CDARP Algorithm FINDANDPASTE finds a solution of cost at most 3 OPT. In fact, we are going to show a stronger result, namelyt that the cost of the solution found by FINDANDPASTE is at most thrice the value of a *lower bound* on OPT.

Definition 4.1 (Flow Bound)**.** For edge $e = [v, v+1]$ of G we define

$$\lambda[v, v+1] := \max\left\{ \lceil \mu_v(R^{\uparrow})/C \rceil, \lceil \mu_v(R^{\downarrow})/C \rceil, 1 \right\}$$

The value

$$C_{\text{flow}} := 2 \sum_{1 \le v \le n-1} \lambda[v, v+1] \cdot d(v, v+1)$$

is called the *flow bound.*

Notice that $\max\{\lceil \mu_v(R^{\uparrow})/C \rceil, 1\}$ is a lower bound on the number of times any feasible transportation must traverse edge $[v, v+1]$ in the direction from vertex v to $v+1$. Similarly, $\max\{\lceil \mu_v(R^{\downarrow})/C \rceil, 1\}$ is a lower bound any transportation must traverse $[v, v+1]$ in direction from $v+1$ to v. Since any feasible transportation always returns to the start point, this implies that in fact the flow bound C_{flow} is a lower bound on the optimal solution cost:

Lemma 4.2. *For any instance I of* CDARP *it follows that* $\mathsf{OPT}(I) \ge C_{\mathit{flow}}$. □

One ingredient for bounding the cost of the solution found by FINDANDPASTE lies in a closer look at the algorithm from [AK88] for solving DARP (CDARP with capacity $C = 1$) in polynomial time on paths. We are going to describe this algorithm and point out the crucial details which will be used in the sequel.

Let (V, E, A) be a mixed graph given in an instance of DARP where (V, E) is a path. The algorithm [AK88] first "balances" the graph (V, A) by adding additional "balancing" arcs B such that for any edge $[v, v+1]$ the number of upward arcs from $A \cup B$ covering $[v, v+1]$ equals the number of downward arcs from $A \cup B$ covering $[v, v+1]$. This implies that in the graph $(V, A \cup B)$ each vertex $v \in V$ satisfies $\deg^+_{A \cup B}(v) = \deg^-_{A \cup B}(v)$. In a second step the algorithm adds a set C of "connecting arcs" of minimum weight such that $(V, A \cup B \cup C)$ is strongly connected and the degree-balance is maintained. The graph $(V, A \cup B \cup C)$ is Eulerian and it can be can be shown that a Eulerian cycle in $(V, A \cup B \cup C)$ yields an optimum transportation. We refer to [AK88] for details.

We are ready to prove the main result about FINDANDPASTE:

Theorem 4.3. *Algorithm* FINDANDPASTE *finds a solution of cost at most* $3C_{\mathit{flow}}$.

Proof. We show that for a specific edge $[v, v+1]$ the number of upward moves constructed in Step 8 of FINDANDPASTE which traverse $[v, v+1]$ in direction from v to $v+1$ is bounded from above by $2\lambda[v, v+1]$. The bound for the number of downward moves constructed in Step 16 is established analogously.

Denote the set $R^{\uparrow}$ at the beginning of the kth iteration of the **while**-loop enclosing Step 8 by $R^{\uparrow(k)}$. Then $R^{\uparrow(1)} = R^{\uparrow}$ and $\mu_v(R^{\uparrow(1)}) \le \lambda[v, v+1]$ for any $v \in V$.

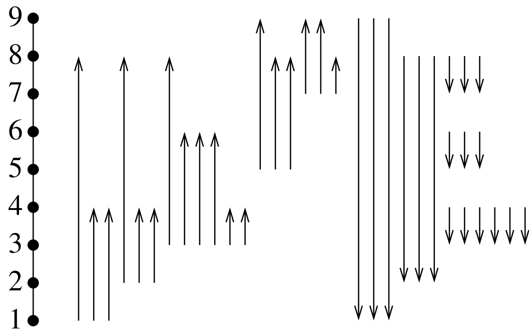

FIGURE 2. An instance of CDARP given by a path G with 9 vertices, and a set of requests R. For a less cluttered display the directed arcs corresponding to R are drawn parallel to the path $G[E]$.

Suppose that edge $[v, v+1]$ is contained in a segment S used in Step 4 in the kth iteration of the **while**-loop. In this case $\mu_v(R^{\uparrow(k)}) > 0$. By a single call to FINDUPARCS μ_v decreases strictly if it is greater than zero since by Corollary 3.6 at least one arc from $M_1 \cup M_2$ covers $[v, v+1]$. Since FINDUPARCS is called C times in iteration k we conclude that $\mu_v(R^{\uparrow(k+1)}) \leq \max\{\mu_v(R^{\uparrow(k+1)}) - C, 0\}$.

Hence $[v, v+1]$ can be used at most $\lceil \mu_v(R^{\uparrow})/C \rceil$ times in a segment in Step 4 and thus by at most $2\lceil \mu_v(R^{\uparrow})/C \rceil \leq 2\lambda[v, v+1]$ upward moves constructed in Step 8. As noted above, the analogous bound for the number of downward moves traversing $[v, v+1]$ is established similarly.

So far we know that for each edge $[v, v+1]$ at most $2\lambda[v, v+1]$ upward moves from Step 8 and at most $2\lambda[v, v+1]$ downward moves from Step 16 traverse $[v, v+1]$. We now consider the chaining of the sequences of moves in Step 20 with the help of the polynomial time algorithm for DARP on paths from [AK88].

Notice that by adding balancing arcs (and corresponding empty moves) the maximum number of moves that traverse $[v, v+1]$ does not increase. Hence, it suffices to consider the empty moves corresponding to connecting arcs C. Since the set of arcs $\{\,(v, v+1), (v+1, v) : 1 \leq v \leq n-1\,\}$ is a feasible set of connecting arcs (which also preserves balance), it follows that the cost of the connecting arcs C chosen by the algorithm from [AK88] can not add empty moves of weight more than $2d(1, n)$. Thus, the total weight of the solution found by Algorithm FINDANDPASTE is not greater than

$$2 \sum_{1 \leq v \leq n-1} \lambda[v, v+1] d(v, v+1) + 2d(1, n) = 2\,C_{\text{flow}} + 2d(1, n) \leq 3\,C_{\text{flow}}.$$

This completes the proof. □

5. EXAMPLE

In this section we illustrate Algorithm FINDANDPASTE on a small example. Consider the instance of CDARP shown in Figure 2. In this instance $G[E]$ is a path with 9 vertices and there are $|R| = 36$ requests to be transported by a server of capacity $C = 3$.

First the requests in $R^{\uparrow}$ are processed. At the beginning the set $R^{\uparrow}$ consists only of one segment $S = R^{\uparrow}$. Now, FINDUPARCS is called $C = 3$ times producing a total

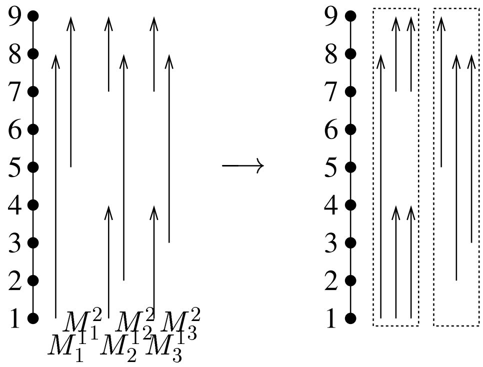

FIGURE 3. The sets $M_1^j, M_2^j, j = 1, 2, 3$ obtained by the first call of FINDUPARCS and the resulting sequences of upward moves U_1 and U_2 constructed by FINDANDPASTE.

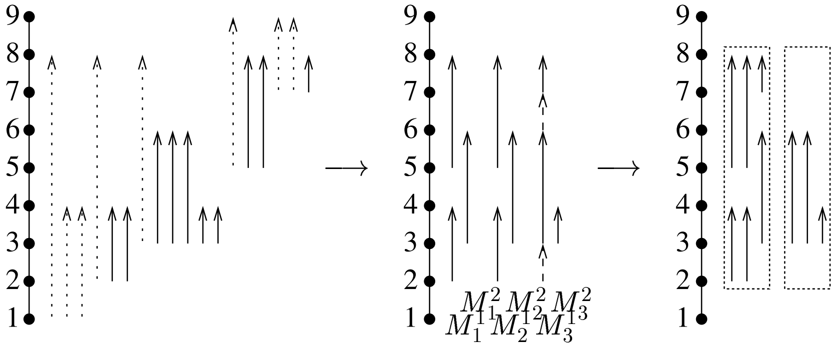

FIGURE 4. The dotted arcs are removed from $R^\uparrow$. Then FINDUPARCS is called again to construct sets of arcs, which are used to build upward moves.

of six sets M_1^j, M_2^j, $j = 1, 2, 3$. These sets and the corresponding upward moves U_1 and U_2 produced from the M_i^j are shown in Figure 3.

All requests transported in the upward moves in U_1 and U_2 are removed from $R^\uparrow$. Since $R^\uparrow \neq \emptyset$, the subroutine FINDUPARCS is called again $C = 3$ times. Notice, that the modified set $R^\uparrow$ still consists of one segment, but now covering only the edges $[v, v+1]$, $v = 2, \ldots, 7$. The result of the second round of calls to FINDUPARCS is illustrated in Figure 4. Notice that in the second round of calls to FINDUPARCS a new situation arises. After the second call in this round, the residual set $R^\uparrow$ does not cover a connected path anymore. Hence, dummy arcs $(2, 3)$ and $(6, 7)$ are added (see Figure 5, the dummy arcs are shown as dotted arcs). The dummy arcs are not included in the sequences constructed in Step 8.

After the second round of calls to FINDUPARCS there is only one more request remainig. This arc will be transported by a single move.

Now, the downward requests $R^\downarrow$ are processed in a similar way with the help of FINDDOWNARCS. This results in the sequences of moves shown in Figure 6.

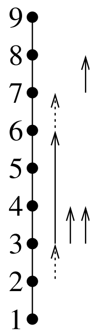

FIGURE 5. Dummy arcs (dotted) added in the second round of calls to FINDUPARCS.

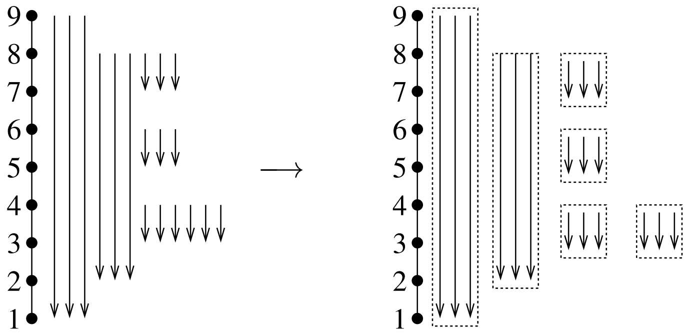

FIGURE 6. The set $R^{\downarrow}$ and some moves transporting all objects in $R^{\downarrow}$.

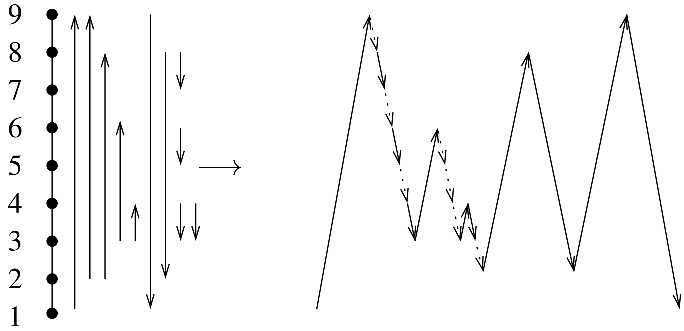

FIGURE 7. The instance Π of DARP and its optimal solution.

Now FINDANDPASTE constructs an instance Π of DARP and uses the algorithm from [AK88] to find an optimal solution T_{Π} for Π. The instance Π and its solution are shown in Figure 7. The dotted arcs in the solution correspond to balancing arcs added by the algorithm.

The final solution found by FINDANDPASTE is displayed in Figure 8.

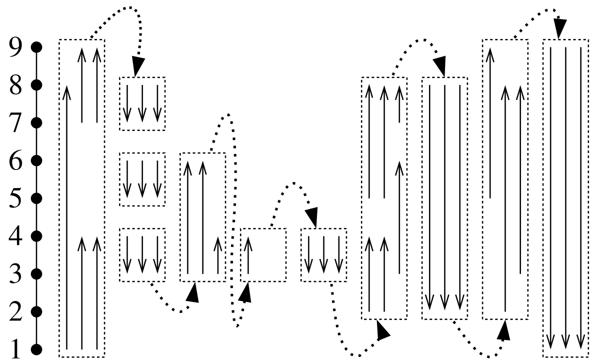

FIGURE 8. The final solution found by FINDANDPASTE. The moves shown as rectangles (and transporting the requests displayed within the rectangles) are executed in the order indicated by the dotted arc sequence.

REFERENCES

[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.

[CR98] M. Charikar and B. Raghavachari, *The finite capacity dial-A-ride problem*, Proceedings of the 39th Annual IEEE Symposium on the Foundations of Computer Science, 1998.

[FG93] G. N. Frederickson and D. J. Guan, *Nonpreemptive ensemble motion planning on a tree*, Journal of Algorithms **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.

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

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

STEFFEN WEIDER, KONRAD-ZUSE-ZENTRUM FÜR INFORMATIONSTECHNIK BERLIN, DEPARTMENT OPTIMIZATION, TAKUSTR. 7, D-14195 BERLIN-DAHLEM, GERMANY.
E-mail address: weider@zib.de