HAL
open science

Relocation in Carsharing Systems using Flows in Time-Expanded Networks

Sven O. Krumke, Alain A. Quilliot, Annegret K. Wagler, Jan-Thierry Wegener

To cite this version:

Sven O. Krumke, Alain A. Quilliot, Annegret K. Wagler, Jan-Thierry Wegener. Relocation in Carsharing Systems using Flows in Time-Expanded Networks. Experimental Algorithms. SEA 2014, 2014, 978-3-319-07958-5. 10.1007/978-3-319-07959-2_8 . hal-00908242

HAL Id: hal-00908242

https://hal.science/hal-00908242

Submitted on 22 Nov 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Relocation in Carsharing Systems using Flows in Time-Expanded Networks

Sven O. Krumke*

Alain Quilliot, Annegret K. Wagler, Jan-Thierry Wegener†

November 21, 2013

Abstract

In a Carsharing System, a fleet of cars is distributed at stations in an urban area, customers can take and return cars at any time and station. For operating such a system in a satisfactory way, the stations have to keep a good ratio between the total number of places and the number of cars in each station, in order to refuse as few customer requests as possible. This leads to the problem of relocating cars between stations, which can be modeled as Pickup and Delivery Problem in a metric space induced by the urban area or, alternatively, by means of flows of cars in convoys in a time-expanded network. We present the resulting integer programming formulations for two variants of the problem: a min-cost flow problem to serve a given set of customer requests at minimal costs, and a max-profit flow problem to additionally solve the decision problem of accepting or rejecting customer requests. These settings can be applied to both the offline and the online version of the relocation problem in order to fully capture the dynamic evolution of the system over time.

1 Introduction

Carsharing is a modern way of car rental, attractive to customers who make only occasional use of a car on demand. Carsharing contributes to sustainable transport as less car intensive means of urban transport, and an increasing number of cities all over the world establish(ed) such services.

In a Carsharing System, a fleet of cars is distributed at specified stations in an urban area, customers can take a car at any time and station and return it at any time and station, provided that there is a car available at the start station and a free place at the final station. To ensure the latter, customers have to book their demands in advance.

For operating such a system in a satisfactory way, the stations have to keep a good ratio between the total number of places and the number of cars in each

*University of Kaiserslautern (Departement of Mathematics), Kaiserslautern, Germany, `krumke@mathematik.uni-kl.de`

†Université Blaise Pascal (Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes), Clermont-Ferrand, France, `{quilliot,wagler,wegener}@isima.fr`. This work was founded by the French National Research Agency, the European Commission (Feder funds) and the Région Auvergne in the Framework of the LabEx IMobS3.

station, in order to refuse as few customer requests as possible. This leads to the problem of balancing the load of the stations, called *Relocation Problem*: an operator has to monitor the load situations of the stations and to decide when and how to move cars from "overfull" stations to "underfull" ones.

Balancing problems of this type occur for any car or bike sharing system, but the scale of the instances, the homogeneity of the fleet, the time delay for prebookings and the possibility to move one or more vehicles in balancing steps differ. We consider an innovative Carsharing System, where the cars are partly autonomous which allows to build convoys of cars, where each convoy is moved by only one driver. This setting is similar to bikesharing, where trucks can simultaneously move several bikes during the relocation process [3, 5, 4]. However, the users of Bikesharing Systems do not book their requests in advance such that the main goal is to guarantee a balanced system during working hours (dynamic situation as in [5]) or to set up an appropriate initial state for the morning (static situation as in [4]). In our case, customers book their requests in advance and we can benefit from this forecast of future (im)balances.

For such a Carsharing System, the load situations of the stations over time can be modeled in terms of a discrete event-based system, and the Relocation Problem can be understood as Pickup and Delivery Problem (PDP) in a metric space encoding the considered urban area [6]. Hereby, a set of routes has to be constructed for the convoys, in order to satisfy transportation requests between "overfull" and "underfull" stations, see Section 2 for details. Problems of this type are known to be hard, see e.g. [1, 2, 7]. In [6], a heuristic approach is presented for the static (offline) version of the Relocation Problem that firstly solves a matching problem to generate transport requests, subsequently solves a PDP, and iteratively augments the transport requests and resulting routes.

Here, we treat the online version of the Relocation Problem in order to capture its dynamic evolution over time. It is natural to interpret the Relocation Problem by means of flows in a time-expanded network, as e.g., proposed by [5] for Bikesharing Systems. In this work, we model the Relocation Problem by coupled flows of cars in convoys in a time-expanded network, taking in addition prebooked customer requests into account, see Section 3 for details. This allows us to treat two variants of the Relocation Problem:

- a min-cost flow problem to determine convoy routes at minimal costs fulfilling a given set of customer requests (quality of service aspect, Section 3.3),
- a max-profit flow problem to decide which customer requests can be satisfied without spending more costs in the relocation process than gaining profit by satisfying customer requests (economic aspect, Section 3.4).

Note that both versions consider flows of cars in convoys, i.e., simultaneous flows of cars and convoy drivers in the same network. Hereby, the two flows are not independent or share arc capacities as in the case of multicommodity flows, but are coupled in this sense that the flow of cars is dependent from the flow of drivers (since cars can only be moved in convoys). The coupling constraints for the two flows reflect the complexity of the studied problem (and the constraint matrix of the resulting integer programming formulations is not totally unimodular as in the case of normal flows).

We close with some remarks on the pros and cons of our approach and future lines of research.

2 Problem Description and Model

In this section, we model the Relocation Problem in the framework of a metric task system.

By [6], the studied Carsharing System can be understood as a discrete event-based system, where

- the system components are the stations $v_1, ..., v_n$, each having an individual capacity $\text{cap}(v_i)$;
- a system state $z^t \in \mathbb{Z}^n$ specifies for each station v_i its load z_i^t at a time point $t \leq T$ within a time horizon $[0, T]$;
- an attribute $\text{att}(v_i, t)$ of station v_i at time t reflects the ratio between its capacity $\text{cap}(v_i)$ and its current load z_i^t, e.g. "overfull" or "underfull";
- states and attributes can be changed by events (customers or convoy drivers take or return cars at a station) or forecasts (customers book requests or the operator generates transport requests).

To balance the load of the stations, an operator has to monitor the system states and to decide when and how to move cars from "overfull" stations to "underfull" ones, in order to fulfill all prebooked demands (Relocation Problem).

For that, the operator monitors the evolution of system states over time, detects imbalanced stations and creates tasks to move cars out of "overfull" stations and into "underfull" ones. More precisely, a *task* is defined by $\tau = (v, rel, due, x)$ where $x \in \mathbb{Z} \setminus \{0\}$ is the number of cars to pickup (if $x > 0$) or to deliver (if $x < 0$) at station v within a time-window $[rel, due] \subseteq [0, T]$ of the *release date rel* and the *due date due*. To fulfill these tasks, routes have to be created for the convoys in order to perform the desired relocation process. For that, it is suitable to encode the urban area where the Carsharing System is running as a *metric space* $M = (V, d)$ induced by a weighted graph $G = (V, E)$ with weight function $w : E \to \mathbb{R}_+$, where the nodes correspond to stations, edges to their physical links in the urban area, and the distance d between two points $v_i, v_j \in V$ is the length of a shortest path from v_i to v_j. In V, we have a distinguished origin $v_o \in V$, the depot of the system.

This together yields a *metric task system*, a pair $(M, \mathcal{T})$ where $M = (V, d)$ is the above metric space and $\mathcal{T}$ a set of tasks, as suitable framework to embed the routes for the convoys.

A truck able to lead a convoy plays the role of a server, the number of available trucks will be denoted by k. Each server has capacity C, corresponding to the maximum possible number of cars per convoy; several servers are necessary to serve a task τ if $x(\tau) > C$ holds.

More precisely, we define the following. Let $f : \mathbb{N} \to \mathbb{N}$ be a function with $f(0) = 0$. An *action* for driver j is a 4-tuple $a = (j, v, f(x), x)$, where $v \in V$ specifies the station $loc(a)$, $f(x)$ is the time $dur(a)$ needed to (un)load $|x|$ cars, and $x \in \mathbb{Z}$ the number of cars $\Delta x(a)$ to be loaded (if $x > 0$) or unloaded (if $x < 0$). Hereby, the capacity of the convoy must not be exceeded, i.e., we have $|x| \leq C$. An action is *empty* if $x = 0$. Note that for an empty action, by definition of f we have $f(x) = 0$. We say that an action is *performed* (by a driver) if he loads (resp. unloads) $|x|$ cars at v.

For technical reasons, the vector $\boldsymbol{z}^t \in \mathbb{N}^{|V|}$ represents the number of cars in a station v at time t before any action is performed.

A *move* from one station to another is a 7-tuple $m = (j, v, t^v, w, t^w, P, x_m)$, where $j \in \{1, \ldots, k\}$ specifies the driver $driv(m)$ that has to move from the *origin station* $orig(m) = v \in V$ starting at time $dep(m) = t^v$ to *destination station* $dest(m) = w \in V$ arriving at time $arr(m) = t^w$, a load of $\ell oad(m) = x_m$ cars in the convoy moving along the path $path(m) = P$. Hereby, we have the following constraints:

(m.i) P is a shortest (v, w)-path,

(m.ii) from $orig(m) \neq dest(m)$ follows $arr(m) = dep(m) + d(orig(m), dest(m))$.

A *tour* for a driver j is a sequence $tour = (m^1, a^1, m^2, a^2, \ldots, a^{n-1}, m^n)$ of moves and actions, with

(t.i) $j = driv(m^1) = driv(a^1) = \cdots = driv(a^{n-1}) = driv(m^n)$,

(t.ii) $dest(m^i) = loc(a^i) = orig(m^{i+1})$,

(t.iii) $arr(m^i) + dur(a^i) = dep(m^{i+1})$,

(t.iv) $0 \leq z^{arr(m^i)}_{dest(m^i)} - \Delta x(a^i) \leq \mathrm{cap}(dest(m^i))$,

(t.v) $\ell oad(m^{i+1}) = \ell oad(m^i) + \Delta x(a^i)$, and,

(t.vi) $0 \leq \ell oad(m^i) + \Delta x(a^i) \leq C$.

For a tour $tour = (m^1, a^1, m^2, a^2, \ldots, a^{n-1}, m^n)$ we define the successive action of a move m^j by $succ^a(m^j) = a^j$. A *transportation schedule* for a metric task system $(M, \mathcal{T})$ is a set of tours $\{tour^1, \ldots, tour^k\}$, such that

(s.i) every driver has exactly one tour,

(s.ii) let $\mathcal{A}$ be the set of all actions then for every task $\tau = (v, rel, due, x) \in \mathcal{T}$ we have
$$\sum_{a \in \mathcal{A}'} \Delta x(a) = x,$$
where $\mathcal{A}' = \{a \in \mathcal{A} \mid a = succ^a(m) \wedge loc(a) = v \wedge rel \leq arr(m) \leq arr(m) + dur(a) \leq due\}$,

(s.iii) for all $(v, t) \in \{(v, t) \in V \times \mathcal{T} \mid (\cdot, v, t, \cdot) \in \mathcal{A}\}$ we have
$$0 \leq z^t_v - \sum_{a \in \mathcal{A}''} \Delta x(a) \leq \mathrm{cap}(v),$$
where $\mathcal{A}'' = \{a \in \mathcal{A} \mid a = succ^a(m) \wedge loc(a) = v \wedge arr(m) = t\}$.

Hereby, (s.ii) ensures that every task in $\mathcal{T}$ is fulfilled by one or more action(s), and (s.iii) ensures that the capacity constraints of all stations are respected, i.e., there are never more than $\mathrm{cap}(v)$ cars parked in any station v.

Example 1 Consider a metric space $M = (V, d)$ induced by graph $G = (V, E)$ from Figure 1. The tasks are induced by customers requesting to take cars from stations at a specific time point and returning them at a later time point to a(nother) station. In this example, a customer requests to take a car from station $\mathcal{E}$ at time 1 and requests to drop it at station $\mathcal{D}$ at time 7, another customer wants to take a car from $\mathcal{C}$ at 1 and to drop it at $\mathcal{E}$ at time 3. Finally, two customers request to take a car from $\mathcal{B}$ at 6, one dropping its car at $\mathcal{A}$ at 7, the other requests to drop the car at $\mathcal{C}$ at time 8. The first two customer requests are fulfilled without relocating cars. However, to fulfill the other two requests, two cars must be transported to the station $\mathcal{B}$, e.g., from $\mathcal{E}$. This leads to the following set of tasks $\mathcal{T} = \{\tau^1 = (\mathcal{E}, 0, 4, 2), \tau^2 = (\mathcal{B}, 0, 6, -2)\}$, i.e., two cars have to be transported from $\mathcal{E}$ to $\mathcal{B}$. The convoys have a capacity $C = 2$. For initial driver positions $\mathcal{A}$ and $\mathcal{D}$, a possible transportation schedule for the metric task system $(M, \mathcal{T})$ is then given by

$$
\begin{aligned}
(1, \mathcal{A}, 0, \mathcal{E}, 1, \{\mathcal{A}, \mathcal{E}\}, 0), &\quad \text{\% move } \mathcal{A} \to \mathcal{E} \\
(1, \mathcal{E}, 0, 1), &\quad \text{\% pickup 1 car (task } \tau^1) \\
(1, \mathcal{E}, 1, \mathcal{E}, 3, \emptyset, 1), &\quad \text{\% wait at } \mathcal{E} \\
(1, \mathcal{E}, 0, 1), &\quad \text{\% pickup 1 car (task } \tau^1) \\
(1, \mathcal{E}, 3, \mathcal{E}, 5, \{\mathcal{E}, \mathcal{A}, \mathcal{B}\}, 2), &\quad \text{\% move } \mathcal{E} \to \mathcal{B} \text{ with 2 cars} \\
(1, \mathcal{B}, 0, -2), &\quad \text{\% deliver 2 cars (task } \tau^2) \\
(2, \mathcal{D}, 0, \mathcal{D}, 5, \emptyset, 0), &\quad \text{\% wait at } \mathcal{E}.
\end{aligned}
$$

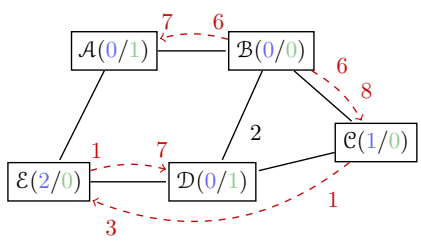

Figure 1: This figure shows a graph G encoding a Carsharing System in an urban area. Each station is drawn in a box. The box contains the name of the station followed by the number of cars and the number of drivers located at the station. Customer requests are drawn as dashed arcs between their pickup and drop stations. The number near the requested pickup station denotes the time point when the customer picks up the car, and analogously, the time point when the customer drops the car is the number near the requested drop station. Each station has a capacity of 3.

The Relocation Problem consists in constructing a transportation schedule from a given metric task system $(M, \mathcal{T})$. The approach in [6] solves this problem in two steps: generate firstly transport requests, secondly solve a PDP. In the next section we propose to solve the Relocation Problem in one step by a min-cost flow problem or a max-profit flow problem to integrate the decision problem of accepting or rejecting customer requests.

3 Flows in Time-Expanded Networks

In this section we propose a way to solve the Relocation Problem in one step by defining a time-expanded network with two coupled flows: a car and a driver flow. The input for the Relocation Problem consists of the following data:

- A network $G = (V, E)$ representing the set of stations V and the road (or logical) connections E between them;
- Driving times $w\colon E \to \mathbb{N}$ encoded by weights on the edges of the network, but no time to load/unload cars from convoys (i.e., we set $f(x) = 0$ for all x in each action);
- Per unit costs cost^{car} and $\text{cost}^{\text{driver}}$ for moving cars and drivers within the network;
- For each station $v \in V$ a capacity $\text{cap}(v) \in \mathbb{N}$ on the number of cars which can be present at this station at any moment in time, the quantities $z^{\text{car}}(v) \leq \text{cap}(v)$ and $z^{\text{driver}}(v)$ of cars and drivers located at v at the start time $t = 0$; the total number of drivers is denoted by $k := \sum_{v \in V} z^{\text{driver}}(v)$;
- The maximum number $C \in \mathbb{N}$ of cars which can be simultaneously moved in a convoy by a single driver;
- A sequence $R = \{r_1, \ldots, r_h\}$ of customer requests arriving over time, where each request has the form $r_j = (p_j, v_j, q_j, w_j)$; $p_j \in \mathbb{N}$ denotes the time when a car is requested to be picked up at station $v_j \in V_S$ and will be dropped at time $q_j \in \mathbb{N}$ at station $w_j \in V_S$. The time p_j is called *requested pickup time*; q_j is called *requested drop time*.
- Per customer request profit $p(r)$ for serving a customer request r.

The output of the Relocation Problem is a transportation schedule for a metric task system, whose tasks are directly induced by the sequence of customer requests. Hereby, we consider two variants: we formulate both a min-cost flow problem to serve all customer requests in R and a max-profit flow problem which rejects customer requests from R whose profit is smaller than the relocation cost to satisfy them.

For that, we build a directed graph $G_T = (V_T, A_T)$, with $A_T = A_H \cup A_L \cup A_R \cup A_S$ as a time-expanded version of the original network G which includes arcs A_R corresponding to the customer requests in R (see Section 3.1).

The cars and drivers will form two flows f^{car} and f^{driver} through G_T which are coupled in the sense that on arcs $a \in A_L$ (the relocation arcs) we have the condition $f^{\text{car}}(a) \leq C \cdot f^{\text{driver}}(a)$ reflecting the dependencies between the two flows. Depending on the setting, we enforce a car flow on all arcs in A_R for the min-cost flow problem (to guarantee quality of service, see Section 3.3) or only on some arcs in A_R (corresponding to the subset of previously accepted requests) for the max-profit flow problem (for economic reasons, see Section 3.4). In both cases, the tasks are directly derived from the sequence R of customer requests.

3.1 Time-Expanded Networks

We build a time-expanded version $G_T = (V_T, A_T)$ of the original network G.

The node set V_T is constructed as follows. Let $\mathcal{T} = \{0, \ldots, T\}$ be a finite set of points in time, where the time horizon T is the maximum drop time of a (yet known) request in R. For each station $v \in V$ and each time point $t \in \mathcal{T}$, there is a node $(v, t) \in V_T$ which represents station v at time t as a capacitated station where cars can be picked up, delivered and exchanged by drivers. Furthermore, it represents the station at time t as a (geographical) transit node for the drivers/convoys. In addition, there is a sink D.

The arc set $A_T = A_H \cup A_L \cup A_R \cup A_S$ of G_T is composed of several subsets:

- For each station $v \in V$ of the original network and each $t \in \{0, 1, \ldots, T-1\}$ there is a *holdover arc* connecting (v, t) to $(v, t+1)$. The set of all holdover arcs is denoted by A_H.
- For each edge (u, v) of G and each point in time $t \in T$ such that $t + d(u, v) \leq T$, there are *relocation arcs* from (u, t) to $(v, t + d(u, v))$; the set of all relocation arcs is denoted by A_L.
- For each customer request $r = (v, t, w, t') \in R$ we add an arc from (v, t) to (w, t'); the set of all customer request arcs is denoted by A_R.
- There are *sink arcs* (denoted by A_S) connecting all nodes $(v, T) \in V_T$ to the sink D.

Note, that by construction the time-expanded network is acyclic.

3.2 Flow Model

On the time-expanded network G_T, we define two different flows, the car flow f^{car} and the driver flow f^{driver}, and specify the capacities as well as the costs for each arc with respect to both flows.

A flow on a relocation arc corresponds to a (sub)move in a tour, i.e., some cars are moved by drivers in a convoy from station u to another station v. Hereby, the stations can be used to pick up or to drop cars, or simply to transit a node (when a driver/convoy passes the station(s) on its way to another station). A relocation arc from (u, t) to $(v, t+d(u, v))$ has infinite capacity for the drivers. However, in order to ensure that each driver moves at most C cars in a convoy and that cars do not move by themselves on such arcs, we require that for all relocation arcs a the inequality

$$f^{\text{car}}(a) \leq C \cdot f^{\text{driver}}(a) \text{ for all } a \in A_R$$

holds. Thus, the capacities for f^{car} on the relocation arcs are not given by constants but by a function. Note that due to these flow coupling constraints, the constraint matrix of the network is not totally unimodular (as in the case of uncoupled flows) and therefore solving such problems becomes hard. Each relocation arc $a = ((u, t), (v, t + d(u, v)))$ corresponding to edge (u, v) has cost

$$c^{\text{car}}(a) := \text{cost}^{\text{car}} \cdot d(u, v) \quad \text{and} \quad c^{\text{driver}}(a) := \text{cost}^{\text{driver}} \cdot d(u, v).$$

A flow on a holdover arc corresponds to cars/drivers remaining at the station in the time interval $[t, t + 1]$. Therefore, the capacity of all holdover arcs with

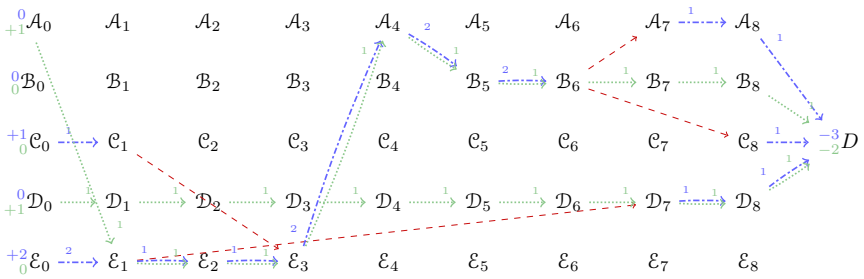

Figure 2: This figure shows an example of a time-expanded network G_T for the graph G from Figure 1. Customer request arcs are shown as dashed arcs. Every node of the form v_t represents a station v at time t. The values $b^{\text{car}}(v,0)$ and $b^{\text{car}}(D)$ are superscripted before the name of the station, and the values $b^{\text{driver}}(v,0)$ and $b^{\text{driver}}(D)$ are subscripted. The driver flow f^{driver} corresponding to transportation schedule from Example 1 is indicated by dotted arcs, and the car flow f^{car} by dash-dotted arcs. The numbers at the arcs correspond to the values of the flows, where the value $f^{\text{car}}(a)$ is shown near the start node of the arc a, and the value $f^{\text{driver}}(a)$ near the end node.

respect to flow f^{car} is set to $\text{cap}(v)$, whereas there is no capacity constraint for f^{driver} on such arcs. Moreover, the cost for both flows on such arcs is zero.

In order to reflect customer requests, we ensure that no driver uses an arc in A_R (by setting its capacity to zero for the driver flow f^{driver}) but set an upper bound of 1 for the car flow f^{car} (and eventually also a lower bound of 1). Request arcs have no costs for both flows (but a profit for the max-profit flow problem).

The sink arcs have infinite capacities with respect to the flow f^{car} and to the flow f^{driver} and no costs.

To correctly initialize the system, we use the nodes $(v,0) \in V_T$ as sources for both flows and set their balances accordingly to the initial numbers of cars and drivers at station v and time 0, i.e.,

$$b^{\text{car}}(v,0) := z^{\text{car}}(v) \quad \text{and} \quad b^{\text{driver}}(v,0) := z^{\text{driver}}(v).$$

For all internal nodes $(v,t) \in V_T$ with $t > 0$, we use normal flow conservation constraints (which is possible due to the fact that the entire flow of cars is modeled by taking both convoy moves and customer actions into account). Finnaly, the sink D has

$$b^{\text{car}}(D) := -\sum_{v \in V} z^{\text{car}}(v) \quad \text{and} \quad b^{\text{driver}}(D) := -\sum_{v \in V} z^{\text{driver}}(v).$$

Figure 2 illustrates a time-expanded network with capacities on the arcs as well as the balances for the nodes $(v,0), D$. Furthermore, both flows f^{car} and f^{car} from Example 1 are shown in this figure. Hereby, every customer request is fulfilled, and thus, the figure shows a feasible solution for the Relocation Problem.

3.3 Integer Linear Program for Min-Cost-Flow

In this subsection, we consider the version of the Relocation Problem that aims at determining convoy routes at minimal costs to fulfill all yet known customer requests from R, i.e., that focuses on the quality of service aspect). For that, we present an integer linear program formulation for a min-cost flow problem in the time-expanded network $G_T = (V_T, A_T)$ as follows:

$$\min \sum_{a \in A_L} c^{\text{car}}(a) f^{\text{car}}(a) + \sum_{a \in A_L} c^{\text{driver}}(a) f^{\text{driver}}(a) \tag{1a}$$

$$\sum_{a \in \delta^-(v,0)} f^{\text{car}}(a) = b^{\text{car}}(v,0) \qquad \forall\ (v,0) \in V_T \tag{1b}$$

$$\sum_{a \in \delta^-(v,0)} f^{\text{driver}}(a) = b^{\text{driver}}(v,0) \qquad \forall\ (v,0) \in V_T \tag{1c}$$

$$\sum_{a \in \delta^-(v,t)} f^{\text{car}}(a) - \sum_{a \in \delta^+(v,t)} f^{\text{car}}(a) = 0 \qquad \forall\ (v,t) \in V_T, t > 0 \tag{1d}$$

$$\sum_{a \in \delta^-(v,t)} f^{\text{driver}}(a) - \sum_{a \in \delta^+(v,t)} f^{\text{driver}}(a) = 0 \qquad \forall\ (v,t) \in V_T, t > 0 \tag{1e}$$

$$\sum_{a \in A_S} f^{\text{car}}(a) = b^{\text{car}}(D) \qquad (v,T) \in V_T \tag{1f}$$

$$\sum_{a \in A_S} f^{\text{driver}}(a) = b^{\text{driver}}(D) \qquad (v,T) \in V_T \tag{1g}$$

$$0 \le f^{\text{car}}(a) \le \text{cap}(v) \qquad \forall\ a = [(v,t),(v,t+1)] \in A_H \tag{1h}$$

$$f^{\text{car}}(a) \le C \cdot f^{\text{driver}}(a) \qquad \forall\ a \in A_L \tag{1i}$$

$$f^{\text{car}}(a) = 1 \qquad \forall\ a \in A_R \tag{1j}$$

$$f^{\text{driver}}(a) = 0 \qquad \forall\ a \in A_R \tag{1k}$$

$$f^{\text{car}}, f^{\text{driver}} \text{ integer,} \tag{1l}$$

where $\delta^-(v,t)$ denotes the set of outgoing arcs of (v,t), and $\delta^+(v,t)$ denotes the set of incoming arcs of (v,t).

The objective function (1a) measures and minimizes the costs of transporting cars in convoys between the stations. Equalities (1b) (resp. (1c)) give the number of cars (resp. drivers) at time 0 for each station. The conditions (1d) and (1e) are the flow conservation constraints for the flows f^{car} and f^{driver}. The equalities (1f) and (1g) ensure that all drivers and cars reach the sink node D. The constraints (1h) give the capacities for f^{car} on holdover arcs which is equal to the capacity of the corresponding station; and the constraints (1i) couple the two flows f^{car} and f^{driver} so that cars on relocation arcs cannot move without drivers. The capacities for the customer request arcs are modeled by equalities (1j) and (1k). Equalities (1j) ensure that each customer request is served, while equalities (1d) ensure that drivers do not use "shortcuts" within the network.

Theorem 2 *Let $(M, \mathcal{T})$ be a metric task system. Then the following three statements are equivalent*

(i) There exists a solution for the Relocation Problem.

(ii) There exist flows f^{driver} and f^{car} satisfying the (in)equalities (1b) *to* (1l).

3.4 Integer Linear Program Max-Flow-Problem

In this section we consider a max-profit flow problem to decide which customer requests can be satisfied without spending more costs in the relocation process than gaining profit by satisfying customer requests. This leads to a Max-Profit Relocation Problem, where the input data specifies, besides all input data from the Relocation Problem, also a profit $p(r)$ for each customer request $r \in R$.

To take the dynamic evolution of the customer requests into account, we partition R into two subsets: R^A of previously accepted customer requests and R^N of newly released customer requests.

The output of the Max-Profit Relocation Problem is a a subset of accepted customer requests $R' \subseteq R$ and a transportation schedule for a metric task system, where the tasks are induced by

- the set of previously accepted customer requests,
- the set of newly released customer requests,
- and the decision which newly released customer request is accepted and thus fulfilled.

A customer request must be either completely fulfilled or completely rejected. In order to achieve this, we adjust the time-expanded network $G_T = (V_T, A_H \cup A_L \cup A_R \cup A_S)$ by considering two subsets $A_{R^A} \cup A_{R^N}$ according to the two request subsets R^A and R^N.

The integer linear program we present for the Max-Profit Relocation Problem is very similar to the one solving the Relocation Problem ((1a) – (1l)). Besides the objective function, we only adjust equalities (1j). Instead of $f^{\text{car}}(a) = 1$ for all customer request arcs, we set $f^{\text{car}}(a) = 1$ for all $a \in A_{R^A}$, to ensure that previously accepted customer requests are fulfilled. For every newly released customer request, we bound the corresponding arc $a \in A_{R^N}$ by $f^{\text{car}}(a) \leq 1$. These inequalities ensure that customer requests can be rejected whenever they have not been accepted a priori, e.g., if the costs for serving the corresponding customer request is more expensive than the profit.

An integer linear program for the Max-Profit Relocation Problem can be given as follows:

$$\max \sum_{a \in A_R} p(a) f^{\text{car}}(a) - \sum_{a \in A_L} c^{\text{car}}(a) f^{\text{car}}(a) - \sum_{a \in A_L} c^{\text{driver}}(a) f^{\text{driver}}(a) \tag{2a}$$

$$\sum_{a \in \delta^-(v,0)} f^{\text{car}}(a) = b^{\text{car}}(v,0) \qquad \forall\ (v,0) \in V_T \tag{2b}$$

$$\sum_{a \in \delta^-(v,0)} f^{\text{driver}}(a) = b^{\text{driver}}(v,0) \qquad \forall\ (v,0) \in V_T \tag{2c}$$

$$\sum_{a \in \delta^-(v,T)} f^{\text{car}}(a) = b^{\text{car}}(D) \tag{2d}$$

$$\sum_{a \in \delta^-(v,T)} f^{\text{driver}}(a) = b^{\text{driver}}(D) \tag{2e}$$

$$\sum_{a\in\delta^-(v,t)} f^{\text{car}}(a) - \sum_{a\in\delta^+(v,t)} f^{\text{car}}(a) = 0 \qquad \forall\ (v,t)\in V_T, t>0 \tag{2f}$$

$$\sum_{a\in\delta^-(v,t)} f^{\text{driver}}(a) - \sum_{a\in\delta^+(v,t)} f^{\text{driver}}(a) = 0 \qquad \forall\ (v,t)\in V_T, t>0 \tag{2g}$$

$$0 \le f^{\text{car}}(a) \le \text{cap}(v) \qquad \forall\ a = [(v,t),(v,t+1)] \in A_H \tag{2h}$$

$$f^{\text{car}}(a) \le C \cdot f^{\text{driver}}(a) \qquad \forall\ a \in A_L \tag{2i}$$

$$f^{\text{car}}(a) \le 1 \qquad \forall\ a \in A_{R^N} \tag{2j}$$

$$f^{\text{car}}(a) = 1 \qquad \forall\ a \in A_{R^A} \tag{2k}$$

$$f^{\text{driver}}(a) = 0 \qquad \forall\ a \in A_{R^A} \cup A_{R^N} \tag{2l}$$

$$f^{\text{car}}, f^{\text{driver}} \text{ integer,} \tag{2m}$$

where $\delta^-(v,t)$ denotes the set of outgoing arcs of (v,t), and $\delta^+(v,t)$ denotes the set of incoming arcs of (v,t).

Theorem 3 *The Max-Profit Relocation Problem has a feasible solution if and only if the Relocation Problem has a feasible solution with R^A as the input sequence of customer requests.*

Example 4 We consider the graph G from Figure 1 and the time-expanded network from Figure 2, respectively. In this example, no customer request is previously accepted, i.e., $R^A = \emptyset$. Depending on the profit given by serving the customer requests $r^3 = (6, \mathcal{B}, 7, \mathcal{A})$ and/or $r^4 = (6, \mathcal{B}, 8, \mathcal{C})$ and the costs of moving cars to station $\mathcal{B}$, e.g., from $\mathcal{E}$, the requests are either served or rejected.

Let us assume that the costs are $\text{cost}^{\text{car}} = 1$ and $\text{cost}^{\text{driver}} = 2$. Then the costs for moving one car by driver 1 (or by driver 2) from $\mathcal{E}$ to $\mathcal{B}$ is 8, and moving two cars is 10. By assuming profits $p(r^3) = 1$ and $p(r^4) = 9$, the profit is maximized when r^4 is served and r^3 is rejected (see Figure 3). Setting $p(r^4) = 7$ results in a rejection of both customer requests, while changing only $p(r^3) = 3$ ensures that both are served. Obviously, similar results can be obtained by changing the costs for the drivers and/or cars.

4 Conclusion

In this paper, we considered the Relocation Problem for Carsharing Systems with semi-autonomous cars. Problems of this type can be modeled as Pickup and Delivery Problem in a metric space induced by the urban area [6] or, alternatively, by means of coupled flows of cars in convoys in a time-expanded network. The here discussed setting is similar to the balancing problem in Bike-sharing Systems (where trucks are used to move several bikes) as in [3-5], but differs in an important aspect: prebooking of customer requests.

We discuss two variants of the Relocation Problem: a min-cost flow problem and a max-profit flow problem. In the min-cost flow problem we determine convoy routes at minimal costs to fulfill all given customer requests (with focus on the aspect of Quality of Service), while in the max-cost flow problem we decide in addition which customer requests can be satisfied without spending

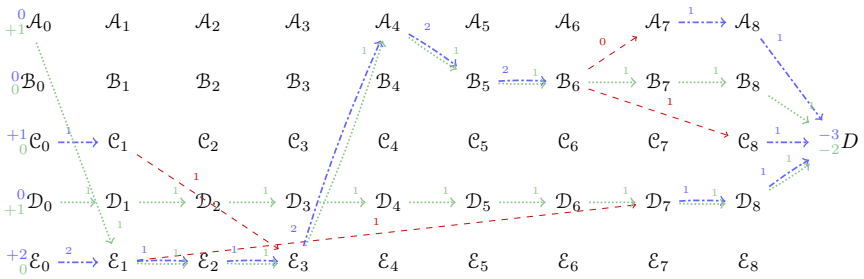

Figure 3: This figure illustrates the Max-Profit Relocation Problem. Hereby, it shows a time-expanded network G_T for the graph G from Figure 1. Customer request arcs are shown as dashed arcs. Every node of the form v_t represents a station v at time t. The values $b^{\text{car}}(v,0)$ and $b^{\text{car}}(D)$ are superscripted before the name of the station, and the values $b^{\text{driver}}(v,0)$ and $b^{\text{driver}}(D)$ are subscripted. The driver flow f^{driver} corresponding to transportation schedule from Example 1 is indicated by dotted arcs, and the car flow f^{car} by dash-dotted arcs. The numbers at the arcs correspond to the values of the flows, where the value $f^{\text{car}}(a)$ is shown near the start node of the arc a, and the value $f^{\text{driver}}(a)$ near the end node. The value on a customer request arc corresponds to whether or not the corresponding request is accepted 1 or rejected 0.

more costs in the relocation process than gaining profit by satisfying customer requests (with focus on the economic aspect).

Unlike in the case of uncoupled flows, the constraint matrix of the time-expanded network is no longer totally unimodular, and reflects the complexity of the Relocation Problem.

For both variants of the Relocation Problem, solving the proposed integer programming formulations yields an exact solution for the *offline situation*: a feasible transportation schedule either serving all customer requests at minimal cost (if it exists) or maximizing the profit (but eventually rejecting certain requests). This is in contrast to the flow formulation proposed in [4] for bike-sharing. There, customers do not book their requests in advance and the goal is to avoid imbalance. For that, two variables per station are considered to reflect their imbalance and the objective is to minimize the total imbalance. As reported in [4], a solution returned by this flow model may require to create or destroy bikes at a station, which leads to an infeasible solution. In contrary, our models keep correctly track of the entire flow of cars (and drivers) in the time-expanded network such that any solution indeed corresponds to a feasible transportation schedule (see Theorem 2 and Theorem 3).

In order to handle the *online situation*, we expect that heuristics based on the two flow formulations can be used. For that, we propose to identify suitable subproblems and to reduce the time-expanded network in two directions:

- omit stations (which are in balance) and relocation arcs which do not carry flows with a certain probability;
- identify an appropriate time interval within the considered time horizon to perform the relocation process and/or use a more coarse time discretization.

The goal is to decrease the expected computation times of the heuristics in such a way that, for instance, the decision to accept or reject one newly released customer request can be given immediately (starting from the previously computed solution for the previously accepted requests).

We further plan to extend the time-expanded network in several ways, modeling different additional aspects of the Relocation Problem. Currently, picking up or dropping a car at a station does not consume any time. Including non-zero durations for such actions in the model is one of our next goals.

Moreover, the time needed for driving from one station to another in an urban area usually depends on the hours of the day. For example, during rush hours, a longer route can be faster than a shorter one, with respect to the distance. Hence, including day time-dependent travel times into our model is also planed for the future.

References

[1] M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors. Network Models, volume 7 of Handbooks in Operations Research and Management Science. Elsevier Science B.V., Amsterdam, 1995.

[2] M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors. Network Routing, volume 8 of Handbooks in Operations Research and Management Science. Elsevier Science B.V., Amsterdam, 1995.

[3] Mike Benchimol, Pascal Benchimol, Benoît Chappert, Arnaud de la Taille, Fabien Laroche, Frédéric Meunier, and Ludovic Robinet. Balancing the stations of a self service "bike hire" system. RAIRO - Operations Research, 45:37–61, 0 2011.

[4] Daniel Chemla, Frédéric Meunier, and Roberto Wolfler Calvo. Bike sharing systems: Solving the static rebalancing problem. pages 120–146, 2013.

[5] C. Contardo, C. Morency, and L-M. Rousseau. Balancing a dynamic public bike-sharing system. Technical Report 9, CIRRELT, 2012. https://www.cirrelt.ca/DocumentsTravail/CIRRELT-2012-09.pdf.

[6] Sven O. Krumke, Alain Quilliot, Annegret Wagler, and Jan-Thierry Wegener. Models and algorithms for carsharing systems and related problems. In Proceedings of the VII Latin-American Algorithms, Graphs, and Optimization Symposium, 2013. To appear in: Electronic Notes of Discrete Mathematics.

[7] G. L. Nemhauser, A. H. G. Rinnooy Kan, and M. J. Todd, editors. Optimization, volume 1 of Handbooks in Operations Research and Management Science. Elsevier Science B.V., Amsterdam, 1989.