Redistribution of Self-service Electric Cars: A Case of Pickup and Delivery

Moshe Dror, Dominique Fortin, Catherine Roucairol

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Redistribution of Self-service Electric Cars: A Case of Pickup and Delivery

Moshe Dror , Dominique Fortin , Catherine Roucairol

No 3543

Novembre 98

THÈME 1

ISSN 0249-6399

INRIA
ROCQUENCOURT

Redistribution of Self-service Electric Cars: A Case of Pickup and Delivery

Moshe Dror * , Dominique Fortin † , Catherine Roucairol ‡

Thème 1 — Réseaux et systèmes
Projet Praxitèle

Rapport de recherche n ° 3543 — Novembre 98 — 14 pages

Abstract: For a large urban area an alternative to public transportation and private cars is considered in the form of a "free" use of electric cars. Such "free" use of electric cars would require periodic redistribution of the cars among different dropoff/pickup stations by a fleet of finite capacity towtrucks stationed at the various depots on the road network. In this study we examine in detail the electric car redistribution problem. The redistribution activity is modelled as that of less than truck load, pickup and delivery with nonsimple paths, allowing for split pickups and deliveries. We propose a Mixed Integer Programming formulation of the problem and we test several solution which include constraint Programming, Lagrangian relaxation methodology applied to the MIP model, and an AI solution approach with a modified A* heuristic. The solution procedures here are designed to solve large practical instances of the car redistribution problem.

Key-words: transportation, electric vehicles, pick up and delivery problems, MIP, A* heuristic

(Résumé : tsvp)

* Management Information Systems Department, College of Business and Public Administration, University of Arizona, Tucson, Arizona, 85721, U.S.A.
† Inria, Domaine de Voluceau, Rocquencourt, B.P. 105, 78153 Le Chesnay Cedex, France.
‡ Laboratoire PRiSM, Université de Versailles-St-Quentin-en-Yvelines, 45, av. des Etats-Unis, 78035 Versailles Cedex, France

Redistribution de véhicules électriques en libre service: un cas de collecte et livraison[§]

Résumé :

Pour une métropole, une alternative au transport public et à la voiture privée peut être la mise en libre service de véhicules électriques. Ce système nécessite une redistribution périodique de véhicules entre les stations en pénurie ou surplus par une flotte de camions de capacité finie stationnée en différents dépôts dans le réseau. Dans cet article, nous présentons le problème et étudions des méthodes de résolution . La redistribution consiste en un problème de tournée avec des chemins non simples et des possibilités de fractionnement des collectes ou livraisons. Nous proposons une modélisation sous forme de programme entier mixte et, comme les solvers de Programmation Par Contraintes ou de Programmation Linéaire ne permettent que de travailler sur des exemples de petite taille, nous donnons une heuristique à la A* pour résoudre notre problème.

Mots-clé : transport, véhicules électriques, pick up and delivery, programmes mixtes en nombres entiers , heuristique A*

[§] This research has been supported by Nato Collaborative Research Grants Programme CRG 971490 and was done while the first and third authors were visiting INRIA.

Contents

1 Motivation and Problem description

French public transportation systems, especially those operating in large urban concentrations such as Paris (Ile de France) have long been concerned with the increasing problems of air pollution and traffic bottlenecks caused by the present private and mass transportation technologies. In light of this concern, an auxiliary transportation mode in the form of electric cars has been developed in the attempt to stem the traffic related pollution levels and alleviate the problems caused by the use of private cars in such urban settings adding the logistics of urban parking to those already mentioned. The idea behind this system is that people would choose to use "free" (freely available and at no cost) electric cars for short distance in-city transportation reducing the demand for mass transit and the use of private conventionally powered vehicles.

The electric cars would be parked at a large number of stations located throughout the metropolitan region, and the users would take a car from one station and leave it at some other station. In order to prevent a station from running out of cars (or of parking space) the electric car system has to provide for a car redistribution mechanism. Such a car redistribution system would be provided by a fleet of tow-trucks positioned at a fixed number of depots in the service region. However, the tow-trucks are of finite capacity (single digit) and their operation of collecting the electric cars and redistributing them appropriately has to be carefully planned in order to minimize waste and provide for a satisfactory level of service. The objective of this study is to analyze this redistribution planning operation and to provide the system's management with an efficient tool for constructing (generating) the best pickup and delivery routes for the fleet of tow-trucks.

This public transport system is actually experimented in France, at Saint-Quentin-en-Yvelines (near Versailles) with a fleet of 50 self-service electric cars (Renault Clio) and 5 parking lots. Two large government research institutes, one in transportation technologies INRETS, the other in Computer Science and Automation INRIA (Praxitele project) work on that project in cooperation with industrial companies (Renault, EDF, Dassault and CGFTE).

The paper is organized as follows. Section 1 gives an abstract description of the problem, assuming that the underlying graph of roads, streets (one way or two way), location for the electric car stations, and the depots for the tow-trucks are given. Each tow-truck depot is identified with its own fleet of vehicles and each such vehicle has to return "at the end of the day" to the original home depot. We initially assume a very generic problem setting with heterogeneous tow-truck fleet and a single depot location. Subsequently, we examine problem extensions and other problem scenarios. Section 2 proposes a modeling approach for the problem which allows then to represent the problem as a Mixed Integer Program (MIP). Section 3 shows that the problem could be decomposed into two more classical problems by using Lagrangian relaxation. Since solvers based either on Constraint Programming or Linear Programming could only solved small size examples (section 4), a A* heuristic approach is designed for pratical size applications.

2 Notation and Mathematical Description of a Single Depot Problem

Let $G = (N \cup D, A)$ be a graph with

- $N = N^+ \cup N^-$ where N^+ is a set of nodes with surplus demand (the source nodes), N^- the set of nodes with demand requests (the sink nodes),
- D the set of depot nodes (initially, we set $D = \{0\}$),
- A is the set of arcs – ordered pairs of nodes– and we assume that all the undirected edges in the graph are represented as two directed arcs.

Q denotes the capacity of the tow-truck and q_i the demand (positive integer in the case of surplus, or negative integer in the case of shortage) at node i. In addition, assume that the cost of traveling between each pair of nodes i and j on the graph is given and denoted by c_{ij}.

Traditionally, in routing formulations a binary decision variable x_{ijv} indicates if the routing solution requires vehicle v to travel directly between point i and point j on the graph. Since split pickups and split deliveries are allowed, already at this point in the problem description we run into some difficulties. In order to illustrate the difficulty of modeling the corresponding pickup and delivery routing problem in this traditional problem description, take the following example.

There are 3 nodes and 1 depot in the graph. The capacity of the vehicle $Q = 5$, and the demands are as follows: $\{q_1 = +6, q_2 = -4, q_3 = -2\}$. We can easily construct a distance matrix for which the optimal solution requires the tow-truck to pickup 5 units at node 1 and deliver 4 of those to node 2 and 1 to node 3, after which the vehicle revisits node 1 to pickup the remaining surplus unit to be delivered to node 3, after which the tow-truck returns to its depot (node 0).

Subsequently, the closed path (circuit) which the vehicle travels is not a simple path (which is an implicit assumption in all traditional vehicle routing formulations with which we are familiar) visiting nodes 1 and 3 two times each.

We will call **Eulerian trail** a circuit which visits each node of the graph at least once except the depot (exactly once).

To alleviate the issue of nonsimple paths in modeling (and solution) of the split pickup and split delivery problem we propose the following pseudopolynomial construction (see figure 1 right) of an **extended graph**.
At each demand point i in the graph with $q_i \geq 2$ (or $q_i \leq -2$) construct a **clique of q_i nodes** connected by arcs of zero weight. Each of the clique nodes is connected to the rest of the graph in the same manner as the original node (note that in terms of data storage and manipulation one can resort to only original arc representation). The new nodes in

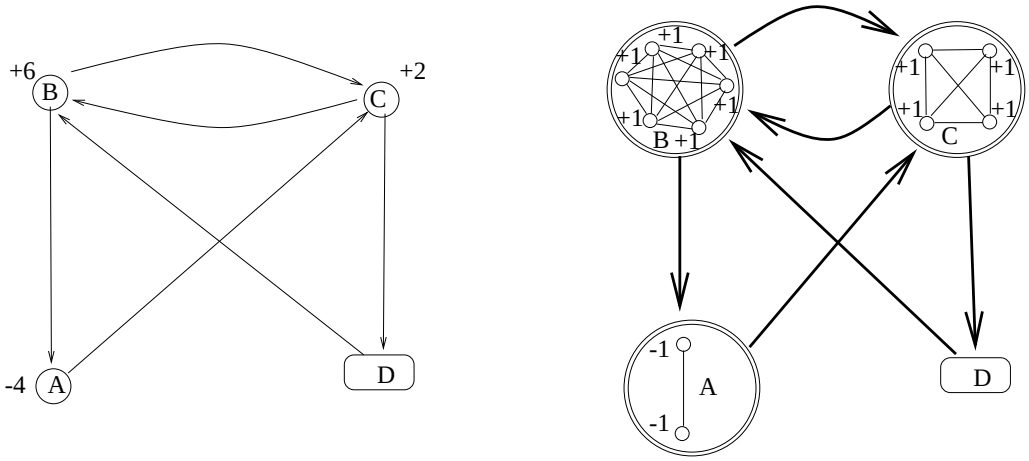

Figure 1: pseudo-polynomial clique expansion

each surplus clique will have demands $q_i = +1$ (or $q_i = -1$ for the shortage clique). Now, in the extended graph we can limit our problem formulation to simple paths.

We assume from now on that our graph G has only demand points of $+1$ or -1 in clique clusters and depot(s).

2.1 Mathematical Formulation for Pickup and Delivery

$$\min \sum_i \sum_j \sum_v c_{ijv} x_{ijv} \tag{1}$$

subject to

$$\begin{aligned}
\sum_i x_{ijv} - \sum_i x_{jiv} &= 0, \quad j = 1, \ldots, n; v = 1, \ldots, NV && (2)\\
\sum_j \sum_v x_{ijv} &= 1, \quad \forall i, && (3)\\
\theta_{ijv} &\leq x_{ijv} Q_v, \quad \forall i, j, v && (4)\\
\sum_i (\theta_{ijv} + x_{ijv}) &= \sum_k \theta_{jkv}, \quad \forall v, j \in N^+, && (5)\\
\sum_i (\theta_{ijv} - x_{ijv}) &= \sum_k \theta_{jkv}, \quad \forall v, j \in N^-, && (6)\\
\sum_j (\theta_{djv} + \theta_{jdv}) &= 0, \quad \forall v, j, d \in D, && (7)\\
\sum_j (x_{djv} + x_{jdv}) &\leq 2, \quad \forall v, j, d \in D, && (8)\\
\sum_{i,j \in S} x_{ijv} &\leq |S| - 1, \quad S \subseteq \{2, \ldots, n\}; \forall v, && (9)\\
x_{ijv} &\in \{0, 1\} && (10)\\
\theta_{ijv} &\in \mathbb{Z}_{\geq} && (11)
\end{aligned}$$

$$\theta_{ijv} \leq Q_v, \quad \forall i,j,v \tag{12}$$

$$\sum_{i,v} \theta_{ijv} \geq 1, \quad j \in N^- \tag{13}$$

$$\sum_{j,v} \theta_{ijv} \geq 1, \quad i \in N^+ \tag{14}$$

NV denotes the upper bound on the number of vehicles and n is the cardinality of the node set without the depot.
x_{ijv} equals 1 if vehicle v goes from i to j and 0 otherwise,
θ_{ijv} represents the number of items (cars in our case) carried by the tow-truck v when going directly from node i to node j.
Constraint sets (2),(5),(6),(7),(8),(13) and (14) are the flow constraints requiring that each vehicle v leaves node i if and only if it enters the node , and leaves and returns to depot d once. Constraint set (3) states that each customer must be assigned to exactly one vehicle. Constraints (4) concerns the capacity of tow-truck v. The number of items carried is restricted by the capacity of the tow-truck Q_v in case of heterogenous fleet composition or just Q if the tow-trucks are of identical capacity.
Constraints (5) and (6) ensures respectively that the load of a vehicle increases (decreases) by one after visiting a surplus (demand) node. Constraint set (6) states that the tow-truck leaves and returns to depot empty.

Note: in the case of non-split pickup and delivery (i.e., if station i is no longer represented by an appropriate size clique and has a given surplus (or shortage of $-q_i$) of $+q_i(\leq Q_v)$, then the tow-truck v servicing i will pickup all of the q_i cars or not serve i), the above formulation is modified by

$$\sum_i (\theta_{ijv} + q_j x_{ijv}) = \sum_k \theta_{jkv}, \forall v, j \in N^+$$

appropriatetly instead of constraints (5) and

$$\sum_i (\theta_{ijv} - q_j x_{ijv}) = \sum_k \theta_{jkv}, \forall v, j \in N^-$$

instead of (6).

Constraint sets (12), (13), (14) are redundant in the above formulation. The three constraints are added here for the sake of the forthcoming Lagrangian relaxation.

In the vehicle routing literature (Fisher, 1994, Nobert, 1982, Laporte et al. 1985) subtour elimination constraints such as (9) above are examined for tightness and compared with subtour elimination constraints below which are equivalent.

$$\sum_{i \in S} \sum_{j \notin S} \sum_v x_{ijv} \geq V(S), \quad S \subset \{1,\ldots,n\}, |S| \geq 2 \tag{15}$$

where $V(S)$ denotes the minimal number of vehicles necessary to serve the set S of customers.

As Fisher (1994) indicates, the choice of the version of subtour elimination constraints is important for the computational experiments and informally, subtour elimination constraints

(9) have been shown to produce tighter bound than (15). Moreover, $V(S)$ corresponds to a bin-packing solution for demands in S. This on itself is a hard problem to solve optimally (Martello and Toth, 1990). However, in the case of unit demands (+1 or -1) the computation of $V(S)$ is trivial.

2.2 Partial Pickup and Delivery

Suppose that on a given day the tow-truck fleet is not expected to fully redistribute the population of electric cars among the stations. Suppose that only a certain percentage α of the cars needs to be redistributed. Still, such partial redistribution has to be planned in a manner which minimizes cost. This again can be mathematically described using the basic optimization model from section 2.1 with some modifications. For completeness, we restate the modified formulation.

$$\min \sum_i \sum_j \sum_v c_{ijv} x_{ijv} \tag{16}$$

subject to

$$\begin{aligned}
\sum_i x_{ijv} - \sum_i x_{jiv} &= 0, \quad j = 1, \ldots, n; v = 1, \ldots, NV, && (17)\\
\sum_j \sum_v x_{ijv} &\leq 1, \quad \forall i, v, && (18)\\
\sum_i \sum_j \sum_v x_{ijv} &\geq \alpha(|N^+| + |N^-|), && (19)\\
\theta_{ijv} &\leq x_{ijv} Q_v, \quad \forall i, j, v && (20)\\
\sum_i (\theta_{ijv} + x_{ijv}) &= \sum_k \theta_{jkv}, \quad \forall v, j \in N^+, && (21)\\
\sum_i (\theta_{ijv} - x_{ijv}) &= \sum_k \theta_{jkv}, \quad \forall v, j \in N^-, && (22)\\
\sum_j (\theta_{djv} + \theta_{jdv}) &= 0, \quad \forall v, j, d \in D, && (23)\\
\sum_j (x_{djv} + x_{jdv}) &\leq 2, \quad \forall v, j, d \in D, && (24)\\
\sum_v \sum_{i \in S} \sum_{j \notin S} x_{ijv} &\geq V(S), \quad |S| \geq 2, S \subseteq \{2, \ldots, n\}, && (25)\\
x_{ijv} &\in \{0, 1\} && (26)\\
\sum_{i,v} \theta_{ijv} &\geq 1, j \in N^- && (27)\\
\sum_{j,v} \theta_{ijv} &\geq 1, i \in N^+ && (28)\\
x_{ijv} &\in \{0, 1\} && (29)\\
\theta_{ijv} &\in \mathbb{Z}_{\geq} && (30)
\end{aligned}$$

Note that the above formulation can also be easily modified if certain (specific) electric car stations need to be only partially replenished.

We can now outline some computational solution approaches to the above models.

3 Solution approaches with solvers

We start by trying to experiment with integer programming solvers based on Linear Programming (Cplex) or Constraints Programming (tools from Ilog Company). The idea was to test if for small size applications that kind of tools could be useful.

The main difficulty with this approach was in addressing the subtour elimination constraints.

3.1 Benchmark problems

The problems have been generated on Paris map. Nine parking stations have been selected:

1. Hôtel de ville,
2. Grand Palais,
3. Gare Montparnasse,
4. Gare d'Austerlitz,
5. Gare du Nord,
6. Tour Eiffel,
7. Bercy,
8. Palais des congrès,
9. Cité Universitaire.

The depot (o) was Gare Saint Lazare. Others parameters could be randomly chosen:

- N set of parking stations (set of nodes) chosen among the 9 preselected parking stations,
- n number of demand (surplus/shortage) nodes,
- q number of electric vehicles,
- q_i quantity of vehicles to be picked /delivered at each demand node i,
- V number of tow-trucks,
- Q_v capacity of each tow-truck.

As the number of variables in the (2)-(14) MIP model is $2n^2V$ and the number of constraints is ??? (without the subtour elimination ones), the choice of the parameters has been limited as follows: N between 3 and 7, q between N and 8, V from 1 to 4, Q less than 5.

3.2 Integer programming solver

For the computational experiments on a *small* problem instances with the MIP model (2)-(14), the subtour elimination constraint (9) (of exponential order in the number of nodes) has been replaced by Miller, Tucker and Zemlin (1960) subtour elimination constrainst which are polynomial with the number of graph nodes(Brochard, 1997).

We introduce new non negative integer variables V_{ijv} which represents the rank in which an arc is visited in the optimal tour.

$$V_{ijv} \leq n x_{ijv}, \quad \forall v, i, j \in N, \tag{31}$$

$$\sum_i (V_{ijv} + x_{ijv}) = \sum_i V_{jiv}, \quad \forall v, j \in N \tag{32}$$

Note that if d is the depot, V_{idv} is in fact the number of nodes visited by v, and so

$$\sum_i V_{idv} = n \ \forall v.$$

For example, a problem with 7 nodes, one depot, 5 vehicles, has 1299 constraints and 880 variables.
Cplex uses a Branch and Bound method based on the continuous relaxation of the problem. Results indicate that resolution time increases with the number of electric cars to redistribute, the number of tow-trucks and their capacity. The number of nodes to be explored in the B&B was limited to 5000; the largest problem exactly solved redistributes 6 vehicles with 3 tow-trucks of capacity 3 and requires 1H30 time for solution. For a greater number of tow trucks (4) and a greater capacity, program stops before exploring all the critical B&B tree, confirming the intuition that this problem may be extremely hard, even with *small* problems.

3.3 Constraint programming tools

Ilog Company provides several tools for object-oriented constraint programming that could be used at hand.
Firstly, we test Ilog Dispatcher which is especially adapted for solving vehicle routing problems. Dispatcher offers a simple object model for representing routing problems in terms of vehicles and visits. The model exploits a library of C++ classes and functions that implement the concepts of vehicle and visits in terms of constrained Solver variables and constraints followed by building solution and a routing plan, The Dispatcher implements a local search method. It offers a variety of predefined heuristics to generate a first solution: nearest addition, nearest insertion, saving heuristics, sweep heuristic. To move from one solution to another, iterative improvement techniques based either on greedy, steepest descent search or Tabu search with move operators (2-opt, Or-opt, relocate, exchange, cross) could be used. Subsequently, the Dispatcher tries to provide to user good solutions but not optimal solutions.

For our problem, we select saving heuristic and Tabu search. Solutions are obtained very quickly on previous examples (less 30 seconds).
In order to assess the quality of these solutions, we use Ilog planner to search for optimal results. Ilog planner is an extension of the Ilog solver (C++ library for object-oriented constraint programming) for solving problems in linear, integer and mixed integer programming. It offers techniques based on the simplex algorithm and Branch and bound.
For our application, we use another formulation of the problem (see below). We search an Eulerian trail in the initial graph (not the extended one). Each time a solution contains a subtour, we introduce the corresponding elimination constrainst. Planner supports very easily that kind of interactive method.

$$\min \sum_i \sum_j \sum_v c_{ijv} x_{ijv} \tag{33}$$

subject to

$$\begin{aligned}
\sum_i x_{ijv} - \sum_i x_{jiv} &= 0, \quad j = 1, \ldots, n; v = 1, \ldots, NV && (34)\\
\sum_j \sum_v x_{ijv} &\geq 1, \quad \forall i, && (35)\\
\theta_{ijv} &\leq x_{ijv} Q_v \forall i, j, v && (36)\\
\sum_k \theta_{jkv} - \sum_i \theta_{ijv} &= q_i, \quad \forall v, j \in N, && (37)\\
\sum_j (\theta_{djv} + \theta_{jdv}) &= 0, \quad \forall v, j, d \in D, && (38)\\
\sum_j (x_{djv} + x_{jdv}) &\leq 2, \quad \forall v, j, d \in D, && (39)\\
\sum_{i \in S} \sum_{j \notin S} x_{ijv} &\geq 1, \quad S \subseteq \{2, \ldots, n\}; \forall v, && (40)\\
x_{ijv} &\in \mathbb{Z}_{\geq}, && (41)\\
\theta_{ijv} &\in \mathbb{Z}_{\geq}. && (42)\\
& && (43)
\end{aligned}$$

x_{ijv} is an integer variable representing the number of times a vehicle v goes from i to j , θ_{ijv} represents the number of electric cars carried by the tow-truck v going from node i to node j.

In figure (2), the depot is close to optimal cost and running time information and each arc is labelled with a couple (#times,global load) of number of times the trail passes through and the corresponding cumulative load while in figure (3), the same example is run a direct search heuritic so each arc is labelled by its actual load.

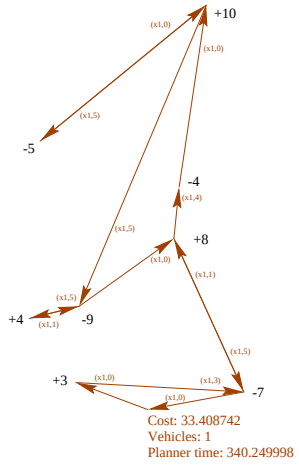

Figure 2: Planner optimal eulerian trail

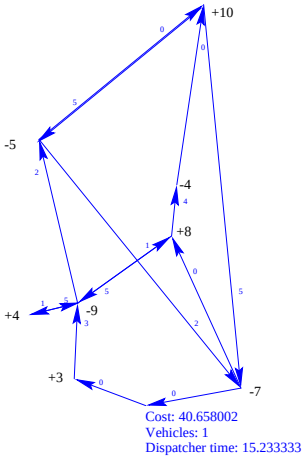

Figure 3: Dispatcher eulerian trail

For all examples of the previous benchmark, optimal solutions have been obtained in about less than 5 minutes. It indicates in fact that these examples are "easy" problems: small number of self-service cars, and parking lots. As an illustration of the increase in difficulty, we give an example specially constructed for INFORMS'98 in Tel Aviv. It takes 15s to obtain an initial solution with Dispatcher and more than 5mn to obtain an optimal solution with Planner for a problem with 1 tow-truck, 25 electric cars, 8 parking lots. The cost optimal solution is 21% better than the initial one. Of course, the number of vehicules to redistribute influenced directly the time taken to find the best solution.

4 A*-heuristic approach

Let us consider our initial problem in the initial graph (without clique expansion). Our goal is to find in G an Eulerian trail of minimal cost with some additional constraints (capacity of tow-truck, demand nodes to satisfy).
We have decided to experiment with an A* type of an algorithm in order to keep the structure of a solution in term of flow and transport. Metaheuristics based on local searches would necessitate more effort to rebuild that kind of structure at each iteration move.
Thus, we revisit the A*-algorithm and explain how to construct a method based upon this A*-algorithm for our redistribution problem.

4.1 An A*-algorithm

An A*-algorithm can be viewed as a generalization of Dijkstra's shortest path algorithm on locally finite graphs from a source stage, say 0, to a target stage $\mathcal{T}$ defined implicitly. Unlike Dijkstra's algorithm, A* remembers only stages in the vicinity of a shortest path from 0 to a target stage, meaning that such a stage has been traversed because it is likely to belong to a shortest path.

Each traversed stage $\mathcal{S}$ is labelled by a pair $(\pi(\mathcal{S}), \phi(\mathcal{S}))$ of values, the first of which is the actual shortest length from the source and the last is the length to a target stage. Since the latter is not known, a third value is introduced such that :

$$\psi(\mathcal{S}) \quad = \quad \pi(\mathcal{S}) + \phi_\epsilon(\mathcal{S}) \tag{44}$$

is a shortest path through $\mathcal{S}$ to a target $\mathcal{T}$,
where $\phi_\epsilon(\mathcal{S})$ is an ϵ-shortest path (epsilon estimated path) from $\mathcal{S}$ to $\mathcal{T}$,
meaning $\phi_\epsilon(\mathcal{S}) \leq (1-\epsilon)\phi(\mathcal{S})$.

Then Dijkstra's algorithm rewrites as :

A*()
{
  // initialize
  $\pi(0) = 0$;
  $\psi(0) = \pi(0) + \phi_\epsilon(0)$;
  frozen $= \emptyset$;
  eligible $= \{0\}$;
  do {
    // select best eligible stage
    $s = \arg \min_{s \in \text{eligible}} \psi(s)$;
    if $\exists \mathcal{T}, \mathcal{T} = s$ {
      // done
      return shortest path to s;
    }
    // proceed outgoing arcs from s
    foreach outgoing arc $s \rightarrow s'$ {
      if $s' \notin$ (frozen$\cup$eligible) or $\pi(s) + l(s,s') + \phi_\epsilon(s') \leq \psi(s')$ {
        // new shortest path through s'

```
            π(s') = π(s) + l(s, s');
            ψ(s') = π(s') + φ_ε(s');
            eligible = eligible ∪{s'};
            frozen = frozen \{s'};
        }
        else {
            frozen = frozen ∪{s'};
        }
    }
  }
}
```

We will refer to A*-algorithm whenever :

(i) ϵ is known ,

(ii) $\forall \mathcal{S} \neq \mathcal{S}', \phi_\epsilon(\mathcal{S}) \leq \phi_{\epsilon'}(\mathcal{S}') + l(\mathcal{S} \rightarrow \mathcal{S}')$

where $l(\mathcal{S} \rightarrow \mathcal{S}')$ is the length of a shortest path between $\mathcal{S}$ and $\mathcal{S}'$.

Note that monotonicity condition (ii) (usually known as consistency property) prevents transition from frozen to eligible set and then minimizes the number of stages examined (in Gondran, Minoux).

When condition (i) is violated, we only achieve (often effective) A*-heuristic, while violation of condition (ii) removes the finiteness property of A*-algorithm.

4.2 An algorithm a la A*

In our case, we define a stage $\mathcal{S} = (\text{demand}, \text{dsptree})$ as a pair: nodes demand and partial directed spanning tree (dsptree).
$\pi(\mathcal{S})$ is the cost of current stage and $\phi_\epsilon(\mathcal{S})$ is an ϵ-optimal cost to turn current stage $\mathcal{S}$ into a goal stage $\mathcal{T} = (\emptyset, \text{minimum cost constrained eulerian trail})$.
Of course, a minimum constrained eulerian trail would yield a (possibly many) solution to our problem.

In practice, ϵ may be hard to compute, however we assume that we could overestimate ϵ as a function of demand $\alpha(\text{demand})$ subject to :

$$\lim_{\text{demand} \rightarrow \emptyset} \alpha(\text{demand}) = 0$$

Below, we concentrate on an implicit computation of $\phi_\epsilon(\mathcal{S})$ in order to improve the likely poor ϵ-approximation based on a TSP on remaining demand nodes (a tour on these nodes clearly underestimates the cost to build a constrained eulerian tour for our problem).
We construct several directed spanning trees by associating some selected *sources* (surplus nodes) to *targets* (shortage nodes). Sources are eligible following some rules defined further. Then, the roots of the directed spanning trees are linked. Finally, we complete the directed spanning tree into an eulerian trail by connecting pending leaves (targets) back to internal

nodes or roots. By insuring that the outdegree of each node equals its indegree, we permit the building of an eulerian trail.

---

computation of $\phi_\epsilon(\mathcal{S})$
{
  // turn current stage into a target one $\mathcal{T}$
  while ($\mathcal{S}$.demand $\neq \emptyset$)
    step($\mathcal{S}$);
  return cost($\mathcal{S}$);
}

---

step($\mathcal{S}$)
{
  // feed directed spanning tree with source→target
  while (eligible **s**→**t**)
    $\mathcal{S}$.update(eligible **s**→**t**);
  /*
   * assert any root demand is non negative
   * assert any internal/leaf demand is non positive
   */

  // link directed spanning trees through source→source
  while (eligible **s**→**s**)
    $\mathcal{S}$.update(eligible **s**→**s**);
  /*
   * assert $\sum_{s \in \mathcal{S}.\text{dsptree}} \text{outdegree}(s) - \text{indegree}(s) = 0$
   */

  // complete the directed spanning tree into an eulerian trail
  // by connecting pending leaves (targets) back to internal nodes
  // (including sources nodes)
  while (eligible **t**→**s**)
    $\mathcal{S}$.update(eligible **t**→**s**);
  /*
   * assert $\sum \mathcal{S}.\text{demand} = 0$
   */
}

Adding the best eligible arc into current stage requires the true add on of the arc into (partial) dsptree as well as the demand update for both source and target nodes (from the computed flow on the arc).
For sake of correctness proof (through assertions in pseudo-code), we avoid mixing of first 2 loops; moreover, it allows to skip over selection details between **s**→**t** and **s**→**s** transitions. We do not emphasize when the depot actually joins the directed spanning tree, since it may occur once in any round of $\phi_\epsilon(\mathcal{S})$ loop.

Our **selection scheme** relies on a mean cost estimation to smooth selection over forecasted demand :

$$\text{eligible } \mathbf{s}\rightarrow\mathbf{t} = \arg\min_{s\rightarrow t}\{\text{meancost}(s\rightarrow t)|\ \forall s'\neq s\ (s'\rightarrow t)\notin \text{dsptree}\} \quad (45)$$

$$\text{eligible } \mathbf{s}\rightarrow\mathbf{s} = \arg\min_{s\rightarrow s'}\{\text{meancost}(s\rightarrow s')|\ s'\in \text{dsptree}\} \quad (46)$$

$$\text{eligible } \mathbf{t}\rightarrow\mathbf{s} = \arg\min_{t\rightarrow s}\{\text{meancost}(t\rightarrow s)|\ s,t\in \text{dsptree}\} \quad (47)$$

Notice that selection rule (45) allows multiple arcs between two nodes in the directed spanning tree without major changes for turning it into an eulerian trail ; in what follows we use directed spanning tree in this weak sense.

$$\begin{aligned}
\text{meancost}(s\rightarrow t) &= \frac{c_{st}}{\min(Q,|q'_s|,|q'_t|)}\\
\text{meancost}(s\rightarrow s') &= \frac{c_{ss'}}{\min(Q,|q'_s|,\text{pushed}(s'))}\\
\text{meancost}(t\rightarrow s) &= \frac{c_{ts}}{\min(|\text{outdegree}(t)-\text{indegree}(t)|,|\text{outdegree}(s)-\text{indegree}(s)|)}
\end{aligned} \quad (48)$$

where q'_s is the current demand in node s.
Finally, in mean cost computation (48) pushed(s') is obtained through the recursive top-down traversal of the directed spanning tree rooted at s' :

```
pushed(s')
{
   p = 0;
   foreach (outgoing arc s' → s)
      p = max(p, min(Q-flow(s' → s),pushed(s)));
   return p;
}
```

where Q-flow($s'\rightarrow s$) is the remaining capacity along this arc. However, some tradeoffs are in order to balance ϵ-approximation against (likely heavy) pushed computation time.

5 Concluding remarks

This article deals with a new routing problem compared to standard TSP or VRP; even with small instances, it happens to be very difficult, due to splitup deliveries and small capacity on the one hand and from the non simple paths constraint on the other hand. On the contrary to local search heuristics, which destroys the flow and transport constraints, an A*-algorithm affords to go around partial eulerian paths to retrieve a good feasible solution.

References

Boone, M., (1994). "A capacitated transportation problem", Report, Laboratoire PRiSM, Université de Versailles-St-Quentin-en-Yvelines, 45, av. des Etats-Unis, 78035 Versailles Cedex, France.

Brochard, J., (1997). "Problème de redistribution des véhicules électriques, Report, Laboratoire PRiSM, Université de Versailles-St-Quentin-en-Yvelines, 45, av. des Etats-Unis, 78035 Versailles Cedex, France.

Chauvet, F., Hafez, N., Proth, J.M. and N., Sauer, (1997), "Management of a pool of self-services cars", INFORMS San Diego, USA.

Dror, M., Fortin, D. and Roucairol, C., (1998), "Redistribution of Electric Cars: A Case of Pickup and Delivery", INFORMS, Tel Aviv, Israel.

Fisher, M.L., (1994). "Optimal solution of vehicle routing problems using minimum K-trees", *Operations Research* 42, 626-642.

Gondran, M. and M., Minoux,(1995), Graphes et algorithmes, 3 rd, Eyrolles.

Goemans, M.X. and Bertsimas, D.J., (1993). "Survivable networks, linear programming relaxations and parsimonious property", *Mathematical Programming* 60, 145-166.

Laporte, G., Nobert, Y., and Desrochers, M., (1985). "Optimal routing under capacity and distance restrictions", *Operations Research* 33, 1050-1073.

Nobert, Y., (1982). "Construction d'Algorithmes Optimaux pour des Extensions au Probleme du Voyageur de Commerce", Doctoral Thesis, Departement d'Informatique et de Recherche Operationnelle, University of Montreal, Canada.

Parent, M., Dumontet, F., Texier, P.Y and Fleurent, F., (1994), "Design and implementation of a public transportation system based on self-service electric cars" IFAC/IFORS. Tianjin. China, 1994.

Unit´e de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399