Dispatching Buses in Parking Depots

Giorgio Gallo       Federico Di Miele

Department of Computer Science, University of Pisa, Corso Italia 40, 56125 Pisa, Italy -

gallo@di.unipi.it

Abstract

Most often, space is the scarce resource in bus depots located in congested urban areas: vehicles arriving at the end of their trips are packed together in a rather small space. This implies that, when a vehicle has to leave to start a new trip, most often, other vehicles have to be moved to clear the way. Here a new model, based on “Min Non-Crossing Matching” and “Generalized Assignment”, is presented to allocate parking spaces to vehicles in a depot in order to minimize the shunting cost. The results obtained applying this approach to a real life case are discussed.

Introduction

Among the many different problems arising in the day-by-day operations of urban transportation companies, the problem of optimally managing the parking space in a vehicle depot has attracted little attention. This problem
is particularly relevant in congested urban areas, where space is a scarce resource: vehicles arriving at the end of their trips are packed together in a rather small space, which implies that, when a vehicle has to leave to start a new trip, most often, other vehicles have to be moved to clear the way. The problem of optimally allocating arriving vehicles to the parking spaces in bus depots will be referred to as the *dispatching problem*.

The dispatching problem is trivial when sufficient space exists in the depot, and parking slots are located in such a way that no parked vehicle can obstruct the movement of any arriving or leaving vehicle. Unfortunately this is not the case in many European towns where depots are small and quite crowded. Even in this case the problem would be easy if all the vehicles were equal, but unfortunately this is not so. Buses are different in size: small buses are required on the lines passing through old city centers, while large buses are needed in rush hours on heavy traffic lines. Some buses may have special lifting devices for disabled people, which are required on certain lines at given hours. Further, buses might carry ads on their sides, and the contracts with the advertisers determine the lines and the hours in which the vehicles should run. Clearly, in this situation, a poor assignment of vehicles to slots at arrival may lead to time consuming maneuvering at departure to clear the way out.

In this paper, after a short description of the problem, we present the solution approach proposed by Winter and Zimmermann (2000), based on a
Quadratic Assignment model. Their model is the only one we know in the literature that captures the main characteristics of our problem. Then we present a new model which, unlike that of Winter and Zimmermann, takes into account the fact that the vehicles have different lengths. This allows a more flexible use of the available parking space, without the need of it being partitioned into slots of equal size as in Winter and Zimmermann’s model. This new model has as main components a set of Generalized Assignment models and a set of Non-Crossing Matching models, and is well suited for a decomposition approach. A decomposition heuristic algorithm is presented together with some computational results on real life problems. An extension of our model that considers the case in which arrivals and departure overlap is also presented.

The paper is organized as follows. Section 1 contains a brief description of the problem. In Section 2 the model by Winter and Zimmermann is described. Sections 3 and 4 contain the description of our model and of the decomposition approach. In Section 5 the algorithm together with some computational results are presented. Section 6 contains a relevant extension of the model. Final remarks follow in the last section.

1 The problem

Here we give a formal description of the Bus Dispatching Problem (BD). First, we assume that the parking space in the depot is partitioned into a
given number of columns of known length; vehicles enter a column at one end and leave it from the other to service the scheduled vehicles duties. Buses are parked in the columns one behind the other. Here, the assumption is made that the operations in the columns mimic the operations in a queue, following a first-in-first-out rule. A last-in-first-out rule could have been chosen instead, so obtaining a stack-like type of operations. The extension to this latter case of the models presented is straightforward. In the following we will denote by \( C = \{1, 2, \ldots, m\} \) the set of the available columns and by \( L_k, k = 1, \ldots, m \), the length of the \( k^{th} \) column. A further assumption is that each vehicle arrives at the depot at night and leaves the day after in the morning. We will drop this assumption in section 6, where the case of vehicles arriving and departing at any time during the day is considered. To avoid confusion, we will use the term ‘vehicle’ only for the arriving vehicles, while, for the leaving ones, we will talk of vehicle duties, or simply duties. In fact, while we know exactly which vehicle arrives at each arrival time, at departure we know only the vehicle duties to be performed: it is part of the problem we are considering to assign vehicles to the duties.

To make the notation simpler, we will identify each vehicle by its index number in the sequence of the arrivals, and each duty by its index number in the sequence of the departures. Thus, \( V = \{1, 2, \ldots, n_v\} \) and \( D = \{1, 2, \ldots, n_d\} \) are the sets of the vehicles and duties, respectively. Although, in general, it may be that \( n_v \neq n_d \), for the sake of simplicity we
assume $n_v = n_d = n$. This assumption is not binding: a number of duties different from $n$ means that either some vehicles will remain in the depot after the last duty has been serviced in the morning, or some vehicles are already present in the depot before the first arrival at night.

We denote by $t'_i$ the type of vehicle $i$, and by $l'_i$ its length; similarly $t''_j$ and $l''_j$ denote the type of vehicle required by duty $j$ and its length, respectively. What we are really interested in is not the particular vehicle arriving or leaving at a given time, but the type of vehicle entering at each arrival time, and the type of vehicle leaving at each departure time.

What we want is to partition the sets $V$ and $D$ into $m$ ordered subsets, say $V_1, \ldots, V_m$, and $D_1, \ldots, D_m$, in such a way that, for each $k \in C$, we have

\begin{align*}
V_k &= \{i_1, i_2, \ldots, i_{n(k)}\}, \\
D_k &= \{j_1, j_2, \ldots, j_{n(k)}\}, \\
i_h &\geq i_{h+1} \text{ and } j_h \geq j_{h+1}, \quad h = 1, \ldots, n(k) - 1, \quad (1) \\
\sum_{h=1}^{n(k)} l'_{i_h} &= \sum_{h=1}^{n(k)} l''_{j_h} \leq L_k. \quad (3)
\end{align*}

The ordering of the elements of the sets $V_k$ corresponds to the order in which the vehicles are parked in the columns, while the ordering of the elements in the sets $D_k$, together with condition (2), ensures that, at each time a vehicle of a certain type is due to leave the depot, there exists a
column with a vehicle of that type available in the first occupied position, i.e., it can leave without the need of moving other parked vehicles. Constraint (3) is a capacity constraint: the total length of the vehicles parked in a column cannot exceed the column length.

It may be the case that no solution satisfying all the constraints exists. For instance if constraints (1) are violated, then there is at least one vehicle that cannot leave at its due time unless one or more parked vehicles are moved. Each time a parked vehicle must be moved to clear the way for a leaving vehicle, we say that there is a crossing. In practice, usually, we are interested in solutions that minimize the number of crossings.

2 A quadratic assignment model

Although problems bearing some resemblance with the Bus Dispatching Problem can be found in the literature (Avriel and Penn, 1995), at the best of our knowledge, the first paper in which BD has been explicitly formulated and solved is due to Winter and Zimmermann (2000).

Winter and Zimmermann present a model that can be viewed as a Quadratic Assignment problem with side constraints. In their model the parking space is partitioned into parking slots of equal size, arranged by columns. This assumption can be, in some cases, rather limiting, mainly when there are many columns, some of small length, and the differences in vehicle lengths are large. This is what happens in some European towns.
where there are small buses for the lines serving the old central districts and larger ones for the lines serving more recent neighborhoods.

In the Winter and Zimmermann model each parking slot is assigned a number, from 1 to \( n \) (the number of parking slots is assumed to be equal to the number of vehicles). Parking slots in the same column have contiguous numbers: namely the parking slots of the first column are 1, 2, \ldots, \( N_1 \), and the slots of the \( k^{th} \) column are \( N_{k-1} + 1 \ldots, N_k \), for \( k = 2, \ldots, m \). The set of all the parking slots is denoted by \( P \). The arriving vehicles are ordered in increasing order of arrival from 1 to \( n \); similarly are ordered the duties.

Introduce coefficients \( \gamma_{ijqp} \) with

\[
\gamma_{ijqp} = \begin{cases} 
1, & \text{if vehicles } i \text{ and } j \text{ cannot be parked in positions } q \text{ and } p, \\
0, & \text{otherwise},
\end{cases}
\]

respectively, without a crossing,

and variables \( x_{iq} \) and \( y_{jq} \) such that

\[
x_{iq} = \begin{cases} 
1, & \text{if the } i^{th} \text{ arriving vehicle is parked in parking slot } q, \\
0, & \text{otherwise},
\end{cases}
\]

\[
y_{jq} = \begin{cases} 
1, & \text{if the vehicle running the } j^{th} \text{ duty is taken from parking slot } q, \\
0, & \text{otherwise}.
\end{cases}
\]

Their model is written as follows:

\[
\text{Minimize } \sum_{i,j \in V} \sum_{q,p \in P} \gamma_{ijqp} x_{jp} x_{iq} \tag{4}
\]
such that

\begin{align}
\sum_{i \in V} x_{iq} &= 1, \quad q \in P, \quad (5) \\
\sum_{q \in P} x_{iq} &= 1, \quad i \in V, \quad (6) \\
\sum_{j \in D} y_{jq} &= 1, \quad q \in P, \quad (7) \\
\sum_{q \in P} y_{jq} &= 1, \quad j \in D, \quad (8) \\
x_{iq} + y_{jq} &\leq 1, \quad i \in V, j \in D, q \in P, t_i' \neq t_j'' \quad (9) \\
y_{jq} + y_{kp} &\leq 1, \quad j, k \in D, q, p \in P, \gamma_{jkqp} = 1, \quad (10) \\
x_{iq} &\in \{0, 1\}, \quad i \in V, q \in P, \quad (11) \\
y_{jq} &\in \{0, 1\}, \quad j \in D, q \in P. \quad (12)
\end{align}

This model assumes that no crossing is permitted at departure, and we arriving vehicles must be parked in such a way that the number of crossings is minimized. Constraints (5) and (6) [(7) and (8)] are matching constraints: each vehicle can occupy exactly one parking slot in the depot. Constraints (9) state that one can assign a duty to a given slot only if the vehicle parked in that parking slot has the right type. Constraints (10) forbid crossings at departure.

To make the problem computationally tractable, the authors have linearized it by

1. adding the constraints

\begin{equation}
\sum_{j \in D} \sum_{p \in P} \gamma_{jqp} x_{jp} + d_{iq} x_{iq} - w_{iq} \leq d_{iq}, \quad i \in V, q \in P,
\end{equation}
where the \( w_{iq} \) are auxiliary variables, and

\[
d_{iq} = \sum_{j \in D} \sum_{p \in P} \gamma_{ijqp}.
\]

2. replacing the objective function with the following

\[
\sum_{i \in V} \sum_{q \in P} w_{iq}.
\]

Furthermore they have introduced additional valid inequalities.

Winter and Zimmermann report on some numerical experimentation made by means of CPLEX 6.5 MIP-Solver: they have been able to solve problems with up to 27 vehicles, 9 column and five different vehicle types.

Since solution times are quite high even for rather small size problems, Winter and Zimmermann have developed heuristics capable of providing acceptable suboptimal solutions at a low computational cost. They also study online algorithms which are very important in this setting. In fact, the actual arrivals of the vehicles may differ substantially from the scheduled ones.

### 3 A two-level model

Here we present a new model which has a two level structure. At one level, we assign arriving vehicles and vehicle duties to columns, while in the second level we try to arrange the vehicles to be parked within the columns in order to satisfy the constraints. We will describe the model presenting first the
submodels for the two levels. We will then combine them to obtain the complete model.

At the first level, arriving vehicles and vehicle duties are assigned to columns in such a way that no column capacity is exceeded. That leads to the following two sets of constraints:

\[
\sum_{k \in C} x_{ik} = 1, \quad i \in V, \quad (13)
\]
\[
\sum_{i \in V} y_{ik} x_{ik} \leq L_k, \quad k \in C, \quad (14)
\]

with

\[
x_{ik} = \begin{cases} 
1, & \text{if vehicle } i \text{ is parked in column } k, \\
0 & \text{otherwise.}
\end{cases}
\]

and

\[
\sum_{k \in C} y_{jk} = 1, \quad j \in D, \quad (15)
\]
\[
\sum_{j \in D} y_{jk} y_{ik} \leq L_k, \quad k \in C, \quad (16)
\]

with

\[
y_{jk} = \begin{cases} 
1, & \text{if vehicle duty } j \text{ is run by a vehicle parked in column } k, \\
0 & \text{otherwise.}
\end{cases}
\]

These are the typical constraints of the generalized assignment problem.

At this level the values of the \(x\) and \(y\) variables are independent from each other, which means that two vectors, \(x\) and \(y\) that satisfy constraints (13) - (16), may result in vehicles parked in a column, which, due to type
mismatch, cannot be used to service the duties assigned to the same column.

To connect the two submodels, that is to link the \( x \) and \( y \) variables, we introduce variables \( z_{ijk} \), with the following meaning:

\[
z_{ijk} = \begin{cases} 
1, & \text{if vehicle duty } j \text{ is run by vehicle } i \text{ parked in column } k, \\
0 & \text{otherwise.}
\end{cases}
\]

Note that these variables are defined only for those pairs \((i, j)\) which are type compatible, i.e., such that vehicle \( i \) can run duty \( j \). In the following we will denote by \( A \) the set of all the compatible \((i, j)\) pairs. Variables \( z \) are subject to the constraints:

\[
\sum_{j \in D(i)} z_{ijk} \leq x_{ik}, \quad i \in V, k \in C, \quad (17)
\]

\[
\sum_{i \in V(j)} z_{ijk} \leq y_{jk}, \quad j \in D, k \in C, \quad (18)
\]

\[
z_{ijk} + z_{uvk} \leq 1, \quad i > u, j < v, (i, j) \in A, (u, v) \in A, k \in C, \quad (19)
\]

where \( V(j) \) is the set of vehicles compatible with duty \( j \) and \( D(i) \) is the set of duties which are compatible with vehicle \( i \), i.e., \( V(j) = \{i \in V : (i, j) \in A\} \) and \( D(i) = \{j \in D : (i, j) \in A\} \).

The meaning of constraints (17) and (18) is straightforward. Constraints (19) forbid crossings in the matchings between parked vehicles and duties: they state that if vehicle \( i \) arrives to the depot after vehicle \( u \) and these vehicles are assigned to the same column, then the duty assigned to \( i \) cannot leave before the duty assigned to \( u \). When this is not the case, i.e., when
duty $j$ has been assigned to vehicle $i$ and duty $v$ has been assigned to vehicle $u$, with $i > u$ and $j < v$, then there is a crossing.

In practice, it may be the case that no crossing free solution exists. This is why, in constraints (17) and (18), inequalities have been used instead of strict equalities: some of the duties may remain unassigned because of the “no crossing” constraint. Once variables $x$ and $y$ have been given a 0 or 1 value, these constraints define the feasible set of $m$ separate non-crossing assignment problems.

Observe that constraints (16) are redundant; in fact they can be easily obtained from (14), (17) and (18), once in these last two constraints the inequalities are replaced by equalities. Still they are important since they strengthen the continuous relaxation of the model.

We can now write the complete model:

\[
(BD) \quad \text{Maximize} \quad \sum_{k \in C} \sum_{(i,j) \in A} z_{ijk} \\
\text{subject to} \quad (13) - (19), \\
\quad x_{ik}, y_{jk}, z_{ijk} \in \{0, 1\}, \quad \forall (i, j, k). \quad (20)
\]

The goal is to assign vehicles and duties to columns in the depot and, at the same time, to try to match the maximum number of duties to vehicles avoiding crossings.

It is possible that in a solution of $BD$, a duty $j$ exists for which $y_{jk} = 1$,
for some $k$, and $\sum_{ik} z_{ijk} = 0$, because no vehicle of the right type exists in column $k$. In this case we say that there is a mismatch, i.e., if we want to implement that solution, the duty will be run by a vehicle of the wrong type.

Although the model does not make any distinction between crossings and mismatches, one can always modify an unfeasible solution, getting a new one containing either only crossings or only mismatches. It is easy to see how mismatches can be traded for crossings to obtain solutions with only mismatches or only crossings (Winter and Zimmermann, 2000).

One may want a solution with either the minimum number of crossing and no mismatches, or a non-crossing solution with the minimum number of mismatches. Our model can be easily modified to obtain such solutions. To produce solutions with minimum number of crossings and no mismatches, it is sufficient to:

1. replace in (17) and (18) the inequalities with equalities, and

2. modify the non-crossing constraint as follows:

$$z_{ijk} + z_{uwk} - w_{ijuk} \leq 1, \quad i > u, j < v, (i, j) \in A, (u, v) \in A, k \in C,$$

where $w_{ijuk}$ is a binary auxiliary variable taking value 1 if and only if the assignments of $i$ to $j$ and $u$ to $v$ in column $k$ give rise to a crossing;

3. minimize the number of crossings, i.e., the function

$$\sum_{k \in C} \sum_{(i,j) \in A} \sum_{(u,v) \in A} w_{ijuk}.$$
To obtain crossing free solution with a minimum number of mismatches it is sufficient to change the inequalities into equalities, as before, and to add variables $z_{ijk}$ also for the non-compatible pairs $(i, j)$. These new variables do not appear in the objective function, so that what is maximized is still the number of compatible matchings between duties and vehicles.

Problem $BD$ is very hard to solve exactly. In fact, it is NP-hard as shown by Winter and Zimmermann (2000). In the next section we will describe a heuristic approach which takes advantage of the problem’s structure and yields good solutions at relatively low cost.

4 A decomposition approach

As observed in the previous section the constraints of problem $BD$ can be partitioned into two main blocks, one corresponding to the non-crossing assignments and the other corresponding to the generalized assignments.

We can relax the problem following a Lagrangean Decomposition approach (Guignard and Kim, 1987). Namely, we

1. replace the variables $x_{ik}$ and $y_{jk}$ in (17) and (18) with the new variables $x'_{ik}$ and $y'_{jk}$;

2. introduce the new constraints $x'_{ik} = x_{ik}, \forall (i, k)$, and $y'_{jk} = y_{jk}, \forall (j, k)$;

3. relax the newly introduced constraints, adding to the objective func-
tion two Lagrangean penalty terms, one for each set of constraints.

The resulting relaxed problem is:

\[
\Phi(\mu, \lambda) = \max_{x_{ik}, y_{jk}} \left\{ \sum_{k} z_{ijk} + \sum_{ik} \lambda_{ik}(x'_{ik} - x_{ik}) + \sum_{jk} \mu_{jk}(y'_{jk} - y_{jk}) \right\},
\]

subject to

\[
\begin{align*}
\sum_{j \in D(i)} z_{ijk} &\leq x'_{ik}, & &i \in V, k \in C, \quad (21) \\
\sum_{i \in V(j)} z_{ijk} &\leq y'_{jk}, & &j \in D, k \in C, \quad (22) \\
z_{ijk} + z_{uvk} &\leq 1, & &i > u, j < v, (i, j) \in A, (u, v) \in A, k \in C, \quad (23) \\
\sum_{k \in C} x_{ik} &\leq 1, & &i \in V, \quad (24) \\
\sum_{i \in V} l'_{xik} &\leq L_{k}, & &k \in C, \quad (25) \\
\sum_{k \in C} y_{jk} &\leq 1, & &j \in D, \quad (26) \\
\sum_{j \in D} l'_{yjk} &\leq L_{k}, & &k \in C, \quad (27)
\end{align*}
\]

\[
x_{ik}, y_{jk}, z_{ijk} \in \{0, 1\}, \quad \forall (i, j, k), \quad (28)
\]

and the corresponding Lagrangean Dual is

\[
(LD) \quad \min_{\mu, \lambda} \{ \Phi(\mu, \lambda) \}.
\]

To evaluate the function \( \Phi(\mu, \lambda) \) we have to solve \( 2m \) generalized assignment problems and \( m \) problems having the property that, once the variables \( x \) and \( y \), which can be considered design variables, have been fixed, the resulting problem becomes a maximum weight non-crossing matching problem. This problem will be called Design Non-Crossing Matching (DNCM). Note
that inequalities can be changed into equalities by adding dummy arcs with zero weight.

The Generalized Assignment problem is well known to be an NP-hard optimization problem. For its solution, we use a mixed integer programming code such as CPLEX. A quite efficient exact ad hoc algorithm for this problem can be found in (Savelsbergh, 1997). $DNCM$ is, however, an easy problem. For each column $k$ we have a $DNCM$ of the type

\[
(DNCM) \text{ Maximize } \sum_{ij \in A} z_{ij} + \sum_{i \in V} \lambda_i x'_i + \sum_{j \in D} \mu_j y'_j
\]

subject to 
\[
\sum_{j \in D(i)} z_{ij} \leq x'_i, \quad i \in V, \quad (29)
\]
\[
\sum_{i \in V(j)} z_{ij} \leq y'_j, \quad j \in D, \quad (30)
\]
\[
z_{ij} + z_{uv} \leq 1, \quad i > u, j < v, (i, j) \in A, (u, v) \in A, \quad (31)
\]
\[
x'_i, y'_j, z_{ij} \in \{0, 1\}, \quad \forall(i, j). \quad (32)
\]

Here to simplify the notation we have dropped the subscript $k$.

It can be shown that $DNCM$ is equivalent to the following weighted non-crossing matching ($WNCM$):

\[
(WNCM) \text{ Maximize } \sum_{ij \in A} (1 + \lambda_i + \mu_j) z_{ij}
\]

subject to 
\[
\sum_{j \in D(i)} z_{ij} \leq 1, \quad i \in V, \quad (33)
\]
\[
\sum_{i \in V(j)} z_{ij} \leq 1, \quad j \in D, \quad (34)
\]
\[
z_{ij} + z_{uv} \leq 1, \quad i > u, j < v, (i, j) \in A, (u, v) \in A, \quad (35)
\]
\[
z_{ij} \in \{0, 1\}, \quad \forall(i, j). \quad (36)
\]
The equivalence is in the sense that, if $\tilde{z}$ is an optimal solution for $WNCM$, then $(\tilde{z}, \tilde{x}, \tilde{y})$ is an optimal solution for $DNQM$, where:

$$\bar{x}_i = \begin{cases} 0 & \text{if } \tilde{z}_{ij} = 0 \quad \forall j, \quad i = 1, \ldots, n, \\ 1 & \text{otherwise,} \end{cases}$$

and

$$\bar{y}_j = \begin{cases} 0 & \text{if } \tilde{z}_{ij} = 0 \quad \forall i, \quad j = 1, \ldots, n, \\ 1 & \text{otherwise,} \end{cases}$$

Problem $WNCM$ has been studied by Malucelli, Ottmann and Pretolani (1993) who have devised an elegant and efficient algorithm for its solution.

This algorithm, which, after the authors, we will call $MOP$, has two phases, a first phase in which a set of labels are determined, and a second phase in which an optimal solution is constructed based on the labels.

**Procedure $MOP (A, \mu, \lambda, M)$:**

begin

for $j := 1$ to $n$ do $L_n[j] := 0$; \{Labelling phase\}

for $i := 1$ to $n$ do

begin

for each $j \in D(i)$ do

$$L_e[i, j] := 1 + \mu_i + \lambda_j + \max \{L_n[k] : k < j\};$$

for each $j \in D(i)$ do

$$L_n[j] := \max \{L_e[i, j], L_n[j]\}$$

end

end
\( M := \emptyset; \bar{M} := A; \) \{Solution construction\}

While \( \bar{M} \neq \emptyset \) do

begin

Choose \((u, v)\) such that \( L_e[u, v] = \max \{L_e[i, j] : (i, j) \in \bar{M}\}; \)

\( \bar{M} := \bar{M} \setminus (u, v); \)

if \( M \cup (u, v) \) is feasible then \( M := M \cup (u, v) \)

end

end.

The set \( M \) returned by \( MOP \) is a maximum weight matching. In the procedure by “\( M \cup (u, v) \) feasible” we mean that \((u, v)\) does not cross any arc already in \( M \). It can be proved that \( MOP \) runs in \( O(n^2 \log n) \) time.

Observe that it also works if \( n_v \neq n_d \), in which case the complexity becomes \( O(n_v n_d \log q) \) time, where \( q = \min \{n_v, n_d\} \). Note that when \( \mu = \lambda = 0 \) the algorithm returns a maximum cardinality non-crossing matching.

5 Solution algorithm

The algorithm follows the following main steps:

1. (Lagrangean Dual). Problem \( LD \) is solved obtaining a solution vector \((\bar{z}, \bar{x}, \bar{x}', \bar{y}, \bar{y}')\); unless \( \bar{x} = \bar{x}' \) and \( \bar{y} = \bar{y}' \), a fortunate but rather unlikely case, the solution is not feasible.
2. (Feasibility). To reach feasibility, we fix variables $x$ and $y$ in $BD$ to the values $\bar{x}$ and $\bar{y}$, respectively, and solve the resulting non-crossing matching problems, obtaining a new vector $\hat{z}$. In general this new solution will leave some vehicles unmatched either because they are of the wrong type or because of the crossings.

3. (Post Optimization). A heuristic procedure is used to match the vehicles and the duties which have not been matched in the previous phases.

The Lagrangean Dual is solved by means of the “Bundle” algorithm described in (Frangioni, 98). This is an ascent algorithm that maximizes the piecewise linear function $\Phi(\lambda, \mu)$, exploiting the information provided by a set (bundle) of subgradients, maintained and updated throughout the iterations performed by the algorithm. Namely, by solving at each iteration a quadratic minimization problem, the algorithm finds a convex combination of the subgradients currently stored and takes it as a candidate ascent direction; a linear search is performed along this direction and, depending on the results, either a move that decreases the objective function is performed or the subgradients’ bundle is enriched so to find a better direction at the next iteration. This approach has been successfully used to solve large scale multicommodity min-cost flow problems (Frangioni and Gallo, 1999).

To describe the heuristic algorithm used in the final phase of our approach we need some further notation. For any $S \subseteq V$ and $T \subseteq D$ define
the set $A(S,T) = \{(i,j) \in A : i \in S, j \in T\}$ and the bipartite graph $G(S,T) = (S,T,A(S,T))$. The algorithm goes through the following steps:

1. **Initialization.** For each column $k$, set $S_k = \{i \in V : \sum_j \hat{z}_{ijk} = 1\}$, and $T_k = \{j \in D : \sum_i \hat{z}_{ijk} = 1\}$. By construction, the set $M_k = \{(i,j) \in A : \hat{z}_{ijk} = 1\}$ is a perfect non-crossing matching for $G(S_k,T_k)$.
   Initialize also the sets $\bar{V} = V \setminus \bigcup_{k \in C} S_k$ and $\bar{D} = D \setminus \bigcup_{k \in C} T_k$.

2. **Non-crossing insertion.** For each $u \in \bar{V}$, each $v \in \bar{D}$ for which $t'_{u} = t''_{v}$, and each $k \in C$ such that $\sum_{i \in S_k} l_i + l_u \leq L_k$, perform the following operations:
   
   (a) if $(u,v)$ does not cross any of the arcs in $M_k$, then set $S_k := S_k \cup \{u\}$, $T_k := T_k \cup \{v\}$, $M_k := M_k \cup \{(u,v)\}$, $\bar{V} := V \setminus \{u\}$, $\bar{D} := D \setminus \{v\}$, and $\hat{z}_{uvk} := 1$.
   
   (b) else, solve a WNCM problem on $G(S_k \cup \{u\}, T_k \cup \{v\})$, and if a non-crossing perfect matching, $\bar{M}$, exists, then set $\hat{z}_{ijk} := 0, \forall (i,j) \in M_k$, $S_k := S_k \cup \{u\}$, $T_k := T_k \cup \{v\}$, $M_k := \bar{M}$, $\bar{V} := V \setminus \{u\}$, $\bar{D} := D \setminus \{v\}$, and $\hat{z}_{ijk} := 1, \forall (i,j) \in M_k$.

3. **Crossing insertion.** While $\bar{D} \neq \emptyset$, select a $u \in \bar{V}$, a $v \in \bar{D}$ for which $t'_{u} = t''_{v}$, a $k$ such that $\sum_{i \in S_k} l_i + l_u \leq L_k$, and set $\bar{V} := V \setminus \{u\}$, $\bar{D} := D \setminus \{v\}$, and $\hat{z}_{uvk} := 1$.

At termination the vector $\hat{z}$ provides a suboptimal solution to the problem.
Table 1 presents the results of an experimentation of the proposed approach on some real cases obtained from the Florence Public Transportation Company. The solution value is given in number of crossings.

The data relative to the use of the Mixed Integer Programming feature of CPLEX have been reported only to provide evidence of the intrinsic difficulty of the problem. In some cases CPLEX was not able to solve the problem within 3 hours (* in the solution column), while in other the problem size exceeded the available memory (** in the corresponding cells).

These results indicate that the decomposition approach provides reasonably good solutions at low computational cost. In fact, the number of crossing is quite low; only in 2 cases it exceeds 10% of the vehicles.

6 An extension of the model

In the model described so far we have assumed a separation between the arrivals and the departures: the first departure starts after the last arrival. While in most case, at least in urban transportation, it can be considered a fairly good approximation of the reality, this is not what happens in practice. Due to the uneven distribution of the service during the day, there are buses arriving at the depot and leaving it more that once. This makes constraints (14) and (16) unnecessarily tight. In fact what we want is that at any point in time the capacity of a given column should not be exceeded, even though the total length of the vehicles using a given column during the all day may
<table>
<thead>
<tr>
<th>Vehicle Types</th>
<th>m</th>
<th>n</th>
<th># of Constraints</th>
<th># of Variables</th>
<th>Cpu Time (sec.)</th>
<th>Solution Value</th>
<th>Cpu Time (sec.)</th>
<th>Solution Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>9</td>
<td>50</td>
<td>4,755,813</td>
<td>13,680</td>
<td>&gt;10,800</td>
<td>*</td>
<td>213.43</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>28</td>
<td>367,150</td>
<td>3,311</td>
<td>&gt;10,800</td>
<td>*</td>
<td>2.28</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>48</td>
<td>3,604,554</td>
<td>12,600</td>
<td>&gt;10,800</td>
<td>*</td>
<td>19.65</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>49</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>141.71</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>31</td>
<td>367,227</td>
<td>3,717</td>
<td>&gt;10,800</td>
<td>**</td>
<td>6.12</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>19</td>
<td>21,690</td>
<td>663</td>
<td>447.70</td>
<td>0</td>
<td>0.06</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>71</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>311.15</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>77</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>**</td>
<td>250.24</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>10</td>
<td>5,632</td>
<td>124</td>
<td>3.15</td>
<td>0</td>
<td>0.01</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>16</td>
<td>13,536</td>
<td>624</td>
<td>48.64</td>
<td>1</td>
<td>0.12</td>
<td>1</td>
</tr>
</tbody>
</table>
exceed it. To solve this problem we have to replace constraints (14) and (16) with a more sophisticated set of constraints.

Two different sets of constraints must be satisfied. The first states that, at any point in time, the total length of the arriving vehicles that have been parked in a column, less the total length of the vehicles that have already left that column, must be less than or equal to the column length.

Let $\tau_i'$ be the time of arrival $i$, $i \in V$, $\tau_j''$ be the time of departure $j$, $j \in D$, and $P(j) = \{i : \tau_i' < \tau_j''\}$ be the set of all the arrivals which are due strictly before departure $j$. Then, the first set of constraints can be written as

$$\sum_{i \in P(h)} l'_i x_{ik} - \sum_{j < h} l''_j y_{jk} \leq L_k, \quad h \in D, k \in C. \quad (37)$$

The second set of constraints states that, at any point in time, the total length of the vehicles that have already departed cannot exceed the total length of parked vehicles. These constraints can be written as

$$\sum_{i \in P(h)} l'_i x_{ik} - \sum_{j \leq h} l''_j y_{jk} \geq 0, \quad h \in D, k \in C. \quad (38)$$

An alternative constraint which carries almost, but not exactly, the same information is

$$\sum_{i \in P(h)} x_{ik} - \sum_{j \leq h} y_{jk} \geq 0, \quad h \in D, k \in C, \quad (39)$$

which states that the number of departing vehicles cannot exceed the number of parked vehicles. While they are similar, constraints (38) and (39) are
different since the total lengths of parked and departing vehicles may be different even if their number is the same, and, conversely, to the same number of departing and leaving vehicles different total lengths may correspond.

The resulting model is the same as (BD) except that (14) and (16) are replaced by (37) and either (38) or (39). Here again, one may observe that (38) and (39) are redundant because of the coupling constraints on the $z_{ijk}$; their role is to make the linear relaxation tighter.

7 Conclusions

We have described the Bus Dispatching Problem and introduced a new model which is easily decomposable into two types of subproblems: Generalized Assignments and Design Non-Crossing Matchings. While the Generalized Assignment problem is NP-hard, medium size instances can often be solved to optimality. The Design Non-Crossing Matching has been proved to be equivalent to the Weighted Non-Crossing Matching which is known to be polynomially solvable.

The new model has the advantage that it does not need the assumption that all the vehicles be of the same size, so making possible a more efficient use of the parking space. Furthermore it can easily be extended to the case where there is an overlap between the stream of arriving vehicles and the stream of departing vehicles.

While the main contribution of this paper lies mainly in the model, we
have also proposed a decomposition approach based on the Bundle method to minimize concave piecewise linear functions. The results obtained provide evidence about the viability of the proposed approach.

The generalization of section 6 suggests an interesting line of future research. This generalized model, jointly with the use of reoptimization techniques, may prove to be an interesting alternative to on-line algorithms used in practical situations.

References


F. Malucelli, T. Ottmann, and D. Pretolani, “Efficient labelling al-
