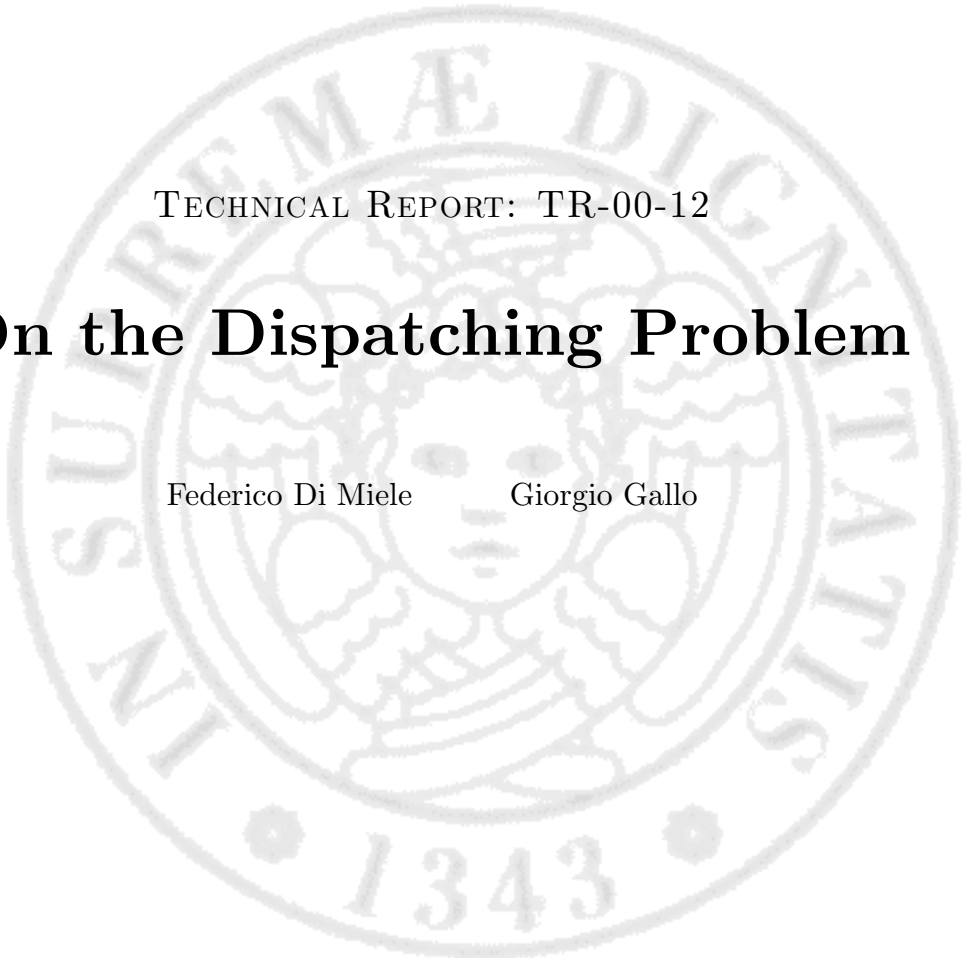
Università di Pisa
Dipartimento di Informatica



Technical Report: TR-00-12

On the Dispatching Problem

Federico Di Miele Giorgio Gallo

October 18, 2000

Address: Corso Italia 40, 56125 Pisa, Italy. Tel: +39 050 887111 Fax: +39 050 887226

On the Dispatching Problem

Federico Di Miele Giorgio Gallo

October 18, 2000

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. That 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 our approach to a real life case are discussed.

1 Introduction

Among the many different problems which arise in the day-by-day operations of urban transportation companies, most of them leading to the formulation of challenging Mathematical Programming problems, one which, at least so far, has attracted rather little attention is the so called "Bus Dispatching Problem". This is the problem of optimally managing the parking space in a vehicle depot.

The dispatching problem is a trivial one when enough space exists in the depot, and parking slots are located in such a way that no parked vehicle can obstacle 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. As an example, in fig. 1 the layout of a real depot located in Florence is shown. Of course, also in this case the problem would be easy if all the vehicles were equal, but unfortunately this is not the case. Busses are different in size: small busses are required on the lines passing through the old center of many medieval cities, while large busses are needed in rush hours on heavy traffic lines. Some busses may have special lifting devices for disabled people, which are required on certain lines at given hours. Further, busses 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.

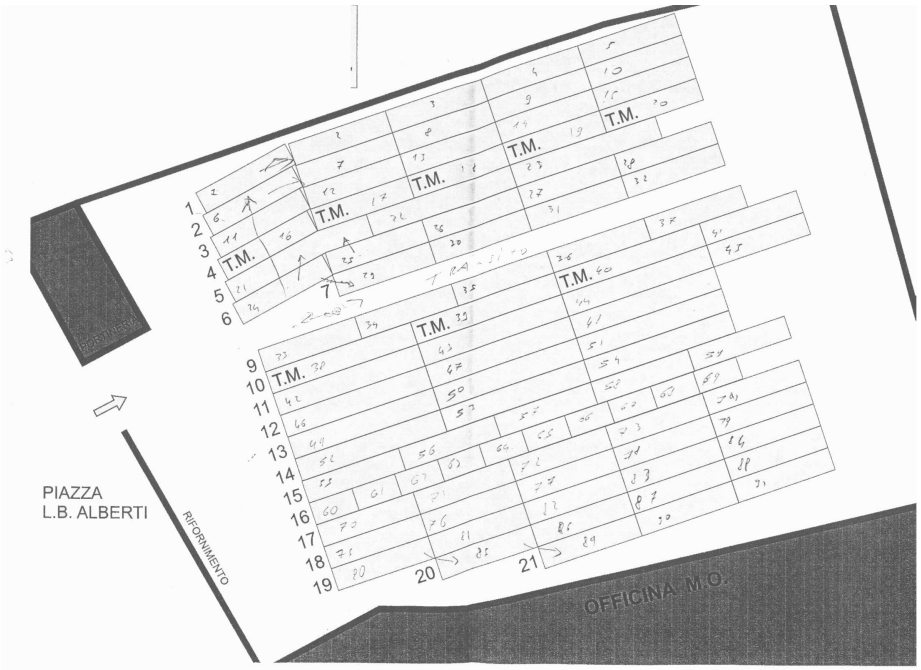


Figure 1: A Depot

In this paper, after a short description of the problem, we present the approach to its solution proposed by Winter and Zimmermann [1], which is based on a Quadratic Assignment model. This model is the only one we know which fully captures the main characteristics of our problem. Then we present a new model which has as main components a set of Generalized Assignment models and a set of NonCrossing Matching models. This new model, which is in our opinion the main contribution of this paper, is well suited for a decomposition approach. An algorithm of this type is presented together with some computational results on real life problems. An extension of our model to take in consideration the case in which arrivals and departure overlap is described before the final section.

2 The Problem

Here we give a formal description of the Bus Dispatching Problem (BDP). First, we assume the parking space in the depot to be 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. busses are parked in the columns one behind the other. Here we have assumed that the operations in the columns mimic the operations in a *queue*. Of course we could have chosen a *stack*-like type of operations; the changes to the models

presented, which are needed to take into account this type of behaviour, are straightforward. In the following we will denote by m the number of columns and by L_i, $i = 1, \ldots, m$, the length of the i^{th} column. A further assumption is that each vehicle arrives to the depot at night and leaves the day after in the morning. We will drop this assumption in section **??**, where the case of vehicles arriving and departing at any time during the day is considered.

To make simpler the notation, in the following we will identify each vehicle with its index number in the sequence of the arrivals, and each duty with 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 the simplicity we assume $n_v = n_d = n$[1].

We will denote by t'_i the type of vehicle i, and by t''_j the type of duty j. Note that what we are really interested in is not the particular vehicle arriving or leaving at a given time, but the type of vehicle which arrives at each arrival time, and the type of vehicle which leaves 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 $k = 1, \ldots, m$ and j, it is

$$
\begin{aligned}
V_k &= \{i_1, i_2, \ldots, i_{n(k)}\}, \\
D_k &= \{j_1, j_2, \ldots, j_{n(k)}\}, \\
i_h &\geq i_{h+1} \text{ and } j_h \geq j_{h+1}, \quad h = 1, \ldots, n(k) - 1, && (1) \\
t'_{i_h} &= t''_{j_h}, \quad h = 1, \ldots, n(k), && (2) \\
\sum_{h=1}^{n(k)} l(t'_{i_h}) &= \sum_{h=1}^{n(k)} l(t''_{j_h}) \leq L_k, && (3)
\end{aligned}
$$

where $l(t)$ is the length of a vehicle of type t.

The ordering on the elements of the sets A_i corresponds to the order in which the vehicles are parked in the columns (vehicles enter their column at one end and move to the other end until they reach the first available position), while the ordering of the elements in the sets D_i, together with condition 2, assures 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 clearing the way moving other parked vehicles. Constraint (3) is a capacity constraint:

[1]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 that some vehicles must be already present in the depot before the first arrival at night.

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 one vehicle at least 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 to a leaving vehicle, we say that there is a *crossing*. In practice, usually, we are interested in solutions which minimize the number of crossings.

3 A Quadratic Assignment model

Although problems bearing some resemblance with the Bus Dispatching Problem can be found in the literature [3][2], at the best of our knowledge, the first paper in which BDP has been explicitly formulated and solved is due to Winter and Zimmermann [1].

Winter and Zimmermann present a model which can be considered as a *Quadratic Assignment* problem with side constraints. In their model the parking space is partitioned into parking spaces of equal size, arranged by columns. Each parking space is assigned a number, from 1 to n (the number of parking spaces is assumed to be equal to the number of vehicles). Parking spaces in the same column have contiguous numbers: namely the parking spaces 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 spaces is denoted by P. The arriving vehicles are ordered in increasing order of arrival from 1 to n; similarly are ordered the leaving vehicles. The model is:

$$\text{Min} \sum_{i,j \in V} \sum_{q,p \in P} \gamma_{ijqp} x_{jp} x_{iq} \tag{4}$$

$$\text{s.t.} \quad \sum_{i \in V} x_{iq} = 1, \qquad q \in P, \tag{5}$$

$$\sum_{q \in P} x_{iq} = 1, \qquad i \in V, \tag{6}$$

$$\sum_{j \in D} y_{jq} = 1, \qquad q \in P, \tag{7}$$

$$\sum_{q \in P} y_{jq} = 1, \qquad j \in D, \tag{8}$$

$$x_{iq} + y_{jq} \leq 1, \qquad i \in V, j \in D, q \in P, t'(i) \neq t''(j) \tag{9}$$

$$y_{jq} + y_{kl} \leq 1, \qquad j, k \in D, q, l \in P, \gamma_{ijlq} = 1, \tag{10}$$

$$x_{iq} \in \{0, 1\} \qquad i \in V, q \in P,$$

$$y_{jq} \in \{0, 1\} \qquad j \in D, q \in P,$$

$$z_{ji} \geq 0, \qquad i \in V, q \in P.$$

where

$$\gamma_{ijqp} = \begin{cases} 1, & \text{if vehicle } i \text{ and } j \text{ cannot be parked in positions } q \text{ and } p, \\ & \text{respectively, without a crossing,} \\ 0, & \text{otherwise.} \end{cases}$$

In this model the assumption has been made that no crossing is permitted at departure, and we want to park the arriving vehicles in such a way that the number of crossings is minimized.

Variable x_{ip} [y_{ip}] is given value 1 iff the i^{th} arriving [leaving] vehicle is parked in [taken from] parking space p; then, constraints (5) and (6) [(7) and (8)] are matching constraints: each vehicle can occupy exactly one parking space in the depot. Constraints (9) say that one can assign a duty to a given slot only if the vehicle parked in that parking space has the right type. Constraints (10) forbid crossings at departure.

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

1. adding the constraint

$$\sum_{j \in D} \sum_{q \in P} \gamma_{ijqp} x_{jp} + d_{iq} x_{iq} + z_{iq} \leq d_{iq}, i \in V, p \in P,$$

   where it is

$$d_{iq} = \sum_{j \in D} \sum_{q \in P} \gamma_{ijqp}$$

2. replacing the objective function with the following

$$\sum_{i \in V} \sum_{q \in P} z_{iq}.$$

Furthermore they have added some redundant constraint which play the role of valid cuts.

Winter and Zimmermann report on some numerical experimentation made by means of Cplex6.5 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 also for rather small size problems, Winter and Zimmermann present also heuristic methods which are capable of providing acceptable suboptimal solution at a low computational cost. They study also online algorithms which are very important in this setting; in fact, the actual arrivals of the vehicles may differ substantially from the scheduled ones.

4 A two-level model

Here we present a new model which has a two level structure. At one level, we assign arriving vehicles and vehicles 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 shall describe the model presenting first the submodels for the two levels and thereafter combining them into the complete model.

At the first level arriving vehicles and vehicle duties are assigned to columns in such a way that no column capacity be exceeded. That leads to the following two sets of constraints, where l_i is the length of the i^{th} vehicle:

$$\sum_{k \in C} x_{ik} = 1, \qquad i = 1, \ldots, n, \tag{11}$$

$$\sum_{i \in V} l_i x_{ik} \leq L_k, \qquad k = 1, \ldots, m, \tag{12}$$

with

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

and

$$\sum_{k \in C} y_{jk} = 1, \qquad j = 1, \ldots, n, \tag{13}$$

$$\sum_{j \in D} l_j y_{jk} \leq L_k, \qquad k = 1, \ldots, m, \tag{14}$$

with

$$y_{ik} = \begin{cases} 1, & \text{if vehicle duty } i \text{ is run by a vehicle parked in column } k, \\ 0 & \text{otherwise.} \end{cases}$$

These are the typical constraints of the *generalized assignment* problem.

At this level the values of the variables x and y are independent from each other, which means that two vectors, x and y which satisfy constraints from (11) to (14), 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 a new set of 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 shall denote by A the set of all the compatible (i, j) pairs.

Variables z are subject to the constraints:

$$\sum_{i \in V(j)} z_{ijk} \leq y_{jk}, \qquad j \in D, k \in C, \tag{15}$$

$$\sum_{j \in D(i)} z_{ijk} \leq x_{ik}, \qquad i \in V, k \in C, \tag{16}$$

$$z_{ijk} + z_{uvk} \leq 1, \qquad i > u, j < v, (i,j) \in A, (u,v) \in A, k \in C, \tag{17}$$

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 (15) and (16) is straightforward. Constraints (17) forbid *crossings* in the matchings between parked vehicles and duties: it says that if vehicle i arrives to the depot before vehicle u and they are assigned to the same column, then the duty assigned to i must leave before the duty assigned to u. In practice, it might be the case that no crossing free solution exists; this is why, in constraints (15) and (16), 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 0/1 value, these constraints define the feasible set of K separate *noncrossing assignment* problems.

Observe that constraints (14) are redundant; in fact they can be easily obtained from (12), (15) and (16), once in these last two constraints the inequalities are replaced by equalities. Still they are important because make the continuous relaxation of our model tighter.

Now, we can write the complete model:

$$\begin{array}{lll} (BD) & \text{Max} & \sum_{k \in C} \sum_{(i,j) \in A} z_{ijk} \\ & \text{s.t.} & (11), (12), (13), (14), (15), (16), (17), \\ & & x_{ik}, y_{jk}, z_{ijk} \in \{0,1\}, \qquad \forall i, j, k. \end{array}$$

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. Note that this model does not make any distinction between crossings and mismatching: a duty j for which $y_{jk} = 1$, for some k, and $\sum_{ik} z_{ijk} = 0$, is a duty assigned to a column in which either no vehicle of the right type exists which can run it, or, if such a vehicle exists, at least one of the parked vehicles has to be moved to clear the way at departure. When no feasible solution exists, one can always get a solution containing either only crossings or only mismatching; it easy to see how a mismatch can be traded for one or possibly more crossings so obtaining solution with only mismatches or only crossings [1].

One may want either a solution with the minimum number of crossing and no mismatching, or a noncrossing solution with the minimum number of

mismatches. Our model can be easily modified to obtain such solutions. To obtain solutions with minimum number of crossings and no mismatching, it is enough to:

1. replace in (15) and (16) the inequalities with equalities;
2. modify the noncrossing constraint as follows:

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

3. minimize the function

$$\sum_{k \in C} \sum_{(i,j) \in A} \sum_{(u,v) \in A} w_{ijuvk}$$

.

To obtain crossing free solution with minimum number of mismatching it is enough to change the inequalities into equalities, as before, and to add variables z_{ijk} also for the noncompatible pairs (i,j). Such new variables do not appear in the objective function, so that what we maximizes is still the number of compatible matchings between duties and vehicles.

Problem BD is very hard to solve exactly. In fact, it is *NPhard* as shown in [1]. In the next section we will describe a heuristic approach which takes advantage of the problem's structure and which can give good solutions at relatively low cost.

5 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 noncrossing assignments and the other corresponding to the generalized assignments.

We can relax the problem following a Lagrangean Decomposition approach, getting the following problem:

$$\begin{aligned}
\Phi(\mu,\lambda) = \quad & \text{Max} \sum_{kij} z_{ijk} + \sum_{jk} \mu_{jk}(y'_{jk} - y_{jk}) + \sum_{ik} \lambda_{ik}(x'_{ik} - x_{ik}) \\
& \sum_{i} z_{ijk} \leq y'_{jk}, \qquad j \in D, k \in C, && (18) \\
& \sum_{j} z_{ijk} \leq x'_{ik}, \qquad i \in V, k \in C, && (19) \\
& z_{ijk} + z_{uvk} \leq 1, \qquad i > u, j < v, k \in C, && (20) \\
& \sum_{k} x_{ik} = 1, \qquad i \in V, && (21) \\
& \sum_{i} l_i x_{ik} \leq L_k, \qquad k \in C, && (22)
\end{aligned}$$

$$\sum_k y_{ik} = 1, \qquad i \in V, \tag{23}$$

$$\sum_i l_i y_{ik} \leq L_k, \qquad k \in C, \tag{24}$$

$$x_{ik}, y_{jk}, z_{ijk} \in \{0,1\}, \qquad \forall i,j,k, \tag{25}$$

and the corresponding Lagrangean Dual

$$(LD) \qquad \text{Max}_{\mu\lambda}\Phi(\mu,\lambda).$$

To simplfy the notation we have left implicit the fact that variables z_{ijk} are defined only for those pairs (i,j) which are compatible, *i.e* $(i,j) \in A$. To derive this relaxation, following a standard procedure, we have changed the name of the x and y variables in the noncrossing assignment problems (x' and y' are the vectors of the new variables), added the constraints $x' = x$ and $y' = y$ and applied Lagrangean relaxation to these new equality constraints.

To evaluate the function $\Phi(\mu,\lambda)$ we have to solve $2K$ generalized assignment problems and K problems which have 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 noncrossing matching problem. This problem will be called Design NonCrossing Matching ($DNCM$). Note that the inequalities can be changed into equalitis by adding dummy arcs with zero weight.

The Generalized Assignment problem is well known to be a NP-hard optimization problem, and, for its solution, we will rely on a Mixed Integer Programming code, such as Cplex. $DNCM$ instead is an easy problem. Let us state formally $DNCM$:

$$\begin{aligned}
(DNCM) \quad \text{Max} \quad & \sum_{ij} z_{ij} + \sum_j \mu_j y'_j + \sum_i \lambda_i x'_i \\
& \sum_i z_{ij} \leq y'_j, \qquad j \in D, \\
& \sum_j z_{ij} \leq x'_i, \qquad i \in V, \\
& z_{ij} + z_{uv} \leq 1, \qquad i > u, j < v.
\end{aligned}$$

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

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

$$\begin{aligned}
(WNCM) \quad \text{Max} \quad & \sum_{ij} (1 + \lambda_i + \mu_j) z_{ij} \\
& \sum_i z_{ij} \leq 1, \qquad j \in D,
\end{aligned}$$

$$\sum_j z_{ij} \leq 1, \qquad i \in V,$$
$$z_{ij} + z_{uv} \leq 1, \qquad i > u, j < v.$$

The equivalence is in the sense that, if $\bar{z}$ is an optimal solution for $WNCM$, then $(\bar{z}, \bar{x}, \bar{y})$ is an optimal solution for $DNCM$, where:

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

and

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

Problem $WNCM$ has been studied by Malucelli, Ottmann and Pretolani [4] who have devised an elegant and efficient algorithm for its solution. Such an 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, μ, λ, M):
begin
for $j := 1$ **to** m **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
$M := \emptyset$; $\bar{M} := C$; {Solution construction}
While $\bar{M} \neq \emptyset$ **do**
begin
(u, v):=argmax$\{L_e[i, j] : (i, j) \in \bar{M}\}$;
$\bar{M} := \bar{M} \setminus (u, v)$;
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 $O(nm \log q)$ time, where $q = \min\{n, m\}$.

Note that when $\mu = \lambda = 0$ the algorithm returns a maximum cardinality noncrossing matching.

Finally, consider the Lagrangean Dual LD. LD reduces to the minimization of a piecewise linear convex function, for which we make use of the *Bundle* type algorithm described in [5]. Mainly, this is an algorithm in which, at each step, a bundle of subgradients, the most recent ones, is used to find an ascent direction; if such a direction appears to be a promising one then a movement along it is performed, otherwise a new subgradient is found and added to the bundle.

Of course, being a relaxation, at least in general, the Dual does not provide a feasible solution. Nevertheless, such a solution can be used as a starting point to get, via some heuristic approach, a good feasible one.

6 Solution algorithm

The solution algorithm we present here will follow 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 get feasibility, we fix variables x and y in BD to the values $\bar{x}$ and $\bar{y}$, respectively, and solve the resulting noncrossing 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 [5]. This is an ascent algorithm which maximizes the piecewise linear function $\Phi(\lambda, \mu)$, exploiting the information provided by a set (bundle) of subgradients which is maintained and updated throughout the iterations performed by the algorithm. Namely, at each iteration, by solving a quadratic minimization problem, the algorithm finds a convex combination of the subgradients currently stored and takes it as a candidate ascent direction; along such a direction a linear search is performed and, depending on the results, either a move which 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 mincost flow problems [6].

To describe the heuristic algorithm used in the final phase of our approach we need some further notation: for any $S \in V$ and $T \in D$ 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:

Table 1: Experimental results

Instance					**Cplex0.6 MIP**		**Decomposition**	
Vehicle Types	m	n	# of Constraints	# of Variables	Cpu Time (sec.)	Solution Value	Cpu Time (sec.)	Solution Value
3	9	50	4,755,813	13,680	>10,800	*	213.43	0
3	7	28	367,150	3,311	>10,800	*	2.28	2
3	9	48	3,604,554	12,600	>10,800	*	19.65	3
4	9	49	**	**	**	**	141.71	3
4	7	31	367,227	3,717	>10,800	**	6.12	6
4	3	19	21,690	663	447.70	0	0.06	0
3	11	71	**	**	**	**	311.15	12
3	12	77	**	**	**	**	250.24	4
2	2	10	5,632	124	3.15	0	0.01	1
2	4	16	13,536	624	48.64	1	0.12	1

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 noncrossing 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. 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 $NCMP$ on $G(S_k \cup \{u\}, T_k \cup \{v\})$, and if a noncrossing perfect matching, $\bar{M}$, exists, then set $S_k := S_k \cup \{u\}$, $T_k := T_k \cup \{v\}$, $M_k := \bar{M}$, $\bar{V} := V \setminus \{u\}$, $\bar{D} := D \setminus \{v\}$, and $\hat{z}_{uvk} := 1$.

3. While $\bar{D} \neq \emptyset$, select a $u \in \bar{V}$, a $v \in \bar{D}$, 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 our problem. Such a solution might contain both *crossings* and *mismatches*.

In Table 1 the results of a preliminary experimentation of the proposed approach on some real cases are presented. For the experimentation have been used four depots located in Florence, each but one for three distinct types of days, working day, Saturday and Sunday.

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 has not be 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).

Although these results are only preliminary, it appears that the decomposition approach may provide reasonably good solutions at low computational cost. In fact, the number of crossing is quite low; only in 2 cases it is exceeds 10% of the vehicles.

7 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. This, although in most case, at least in urban transportation, can be considered a fairly good approximation of the reality, is not what happens in practice. Due to the uneven distribution of the service during the day, there are busses which arrive to the depot and leave from it more that once. That makes constraints (12) and (14) unnecessarily tight: in fact what we want is that at any point in time the capacity of a given column be not exceeded, although the total length of the vehicles using a given column during the all day might exceed it. To solve this problem we have to replace constraints (12) and (14) with a more sophisticated set of constraints.

If τ'_i is the time of arrival i, $i \in V$, and τ''_j is the time of departure j, $j \in D$, then we may define the sets

$$\begin{aligned} P(j) &= \{i : \tau'_i < \tau''_j\}, \\ F(i) &= \{j : \tau'_i < \tau''_j\}; \end{aligned}$$

that is $P(j)$ is the set of all the arrivals which are due strictly before departure j, and $F(i)$ is the set of all the departures which are due strictly after arrival i.

Two different types of constraints must be satisfied. The first states that, at any point in time, the total length of the arriving vehicles which have been parked in a column less the total length of the vehicles which have already left that column must be less than or equal to the column length. This constraint can be written as the following set of inequalities:

$$\sum_{i \in P(h)} l_i x_{ik} - \sum_{j<h} l_j y_{jk} \leq L_k, \quad h \in D, k \in C. \tag{26}$$

The second type of constraints states that, at any point in time, the total length of the vehicles which have already departed cannot exceed the total length of the parked vehicles. It can be written as follows:

$$\sum_{i \in P(h)} l_i x_{ik} - \sum_{j \leq h} l_j y_{jk} \geq 0, \quad h \in D, k \in C. \tag{27}$$

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

$$\sum_{i \in P(h)} x_{ik} - \sum_{j \leq h} y_{jk} \geq 0, \quad h \in D, k \in C, \tag{28}$$

which states that the number of departing vehicles cannot exceed the number of the parked ones. Clearly, although similar, (27) and (28) are different: to the same length for parked and departing vehicles may not correspond an equal number, and, conversely, we may have that the numbers of departing and leaving vehicles are the same while the total lengths are different.

The resulting model is the same as (BD) except that (13) and (14) are replaced by (26) and either (27) or (28). Here again, one may observe that (26), (27) and (28) are redundant because of the coupling constraints on the z_{ijk}; their role is to make the linear relaxation tighter.

8 Conclusions

Here 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 NonCrossing Matchings. While the Generalized Assignment problems can be solved to optimality relatively easily due to the not too large size, the Design NonCrossing Matching has been proved to be equivalent to the Weighted NonCrossing Matching which is known to be polinomially solvable.

The new model has the advantage that 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 be easily extended to the case in which there are overlapping between the stream of arriving vehicles and the stream of departing ones.

Although the main contribution of this paper is intended to be in the model, we have also proposed a decomposition approach based on the Bundle method to maximize concave piecewise linear functions. The results obtained, although still preliminary, provide enough evidence about the viability of this approach.

The generalization of section 7 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 in practical situations.

References

[1] Winter, Zimmermann, "Real-Time Dispatch of Trams in Storage Yards", *Annals of Operations Research* (to appear)

[2] Avriel, Penn, Shpirer, "Container ship stowage problem: complexity and connection to the coloring of circle graphs", submitted

[3] Avriel, Penn, Shpirer, Witteboon, "Stowage planning for container ships to reduce the number of shifts", Annals of Operations Research - Mathematics of Industrial Systems (to appear)

[4] Malucelli, Ottmann, Pretolani, "Efficient labelling algorithms for the maximum noncrossing matching problem", *Discrete Applied Mathematics*, 47 (1993) 175-179

[5] Frangioni A., "Generalized Bundle Methods", T.R. 04/98, Dip. di Informatica, Univ. Pisa, 1998.

[6] Frangioni A., G. Gallo, "A Bundle Type Dual-Ascent Approach to Linear Multicommodity Min-Cost Flow Problems", *INFORMS Journal on Computing*, 11 (1999) 370-393