A Branch and Bound Algorithm for the Multiple Depot Vehicle Scheduling Problem

G. Carpaneto
Dipartimento di Economia Politica, University of Modena, Italy

M. Dell'Amico, M. Fischetti, P. Toth
DEIS, University of Bologna, Italy

The Vehicle Scheduling Problem concerns the assigning of a set of time-tabled trips to vehicles so as to minimize a given cost function. We consider the NP-hard Multiple Depot case in which, in addition, one has to assign vehicles to depots. Different lower bounds based on assignment relaxation and on connectivity constraints are presented and combined in an effective bounding procedure. A strong dominance procedure derived from new dominance criteria is also described. A branch and bound algorithm is finally proposed. Computational results are given.

I. INTRODUCTION

Consider the following *Multiple Depot Vehicle Scheduling Problem* (*MD-VSP*). A set of n time-tabled *trips* $T_1, \ldots, T_n$ is given, the jth starting at time s_j and ending at time e_j ($j = 1, \ldots, n$), as well as m *depots* $D_1, \ldots, D_m$ in the kth of which r_k (with $r_k \leq n$) *vehicles* are stationed ($k = 1, \ldots, m$). All the vehicles are supposed identical. Let $\tau_{i,j}$ be the travel time for a vehicle to go from the ending point of trip T_i to the starting point of trip T_j ($i,j = 1, \ldots, n$). An ordered pair of trips (T_i, T_j) is said to be *compatible* iff the same vehicle can cover trips T_i and T_j in the sequence, i.e., iff $e_i + \tau_{i,j} \leq s_j$. For each compatible pair of trips (T_i, T_j), let $\gamma_{i,j}$ be the finite *cost* incurred if a vehicle performs trip T_j just after trip T_i ($\gamma_{i,j} = \infty$ if (T_i, T_j) is not compatible or $i = j$). For each trip T_j and each depot D_k, let $\bar{\gamma}_{k,j}$ (resp. ${}_{j,k}$) be the finite cost incurred if a vehicle stationed at depot D_k starts (resp. ends) with trip T_j. The cost of a *duty* $(T_{i1}, T_{i2}, \ldots, T_{ik})$ performed by a vehicle stationed at depot D_k is then given by $\bar{\gamma}_{k,i_1} + \gamma_{i_1,i_2} + \cdots + \gamma_{i_{h-1},i_h} + \bar{\gamma}_{i_h,k}$. The problem is to find an assignment of trips to vehicles in such a way that:

i) each trip is covered by exactly one vehicle;

ii) each vehicle used in the solution covers a feasible duty, i.e., a sequence of pairwise compatible trips, and returns to its depot at the end of the duty;

NETWORKS, Vol. 19 (1989) 531–548

CCC 0028-3045/89/050531-18$04.00

iii) the number of vehicles stationed at depot D_k and used in the solution does not exceed r_k ($k = 1, \ldots, m$);

iv) the sum of the costs associated with the duties of the vehicles used in the solution is minimized (the unused vehicles give no contribution to the total cost).

Such a problem arises in the management of transportation companies. Depending on the way the costs are defined, the objective is minimizing, for example:

a) the number of vehicles used in the solution, when $\bar{\gamma}_{k,j} = 1$, $\tilde{\gamma}_{j,k} = 0$ for any trip T_j and any depot D_k, and $\gamma_{i,j} = 0$ for any compatible pair of trips (T_i,T_j);

b) the operational costs, when values $(\gamma_{i,j})$, $(\bar{\gamma}_{k,j})$, and $(\tilde{\gamma}_{j,k})$ are the actual costs of the deadheading transfers of the vehicles (including vehicle and driver costs, idle time, . . .);

c) any combination of a) and b).

When $m \geq 2$, the problem if *NP*-hard (see Bertossi, Carraresi, and Gallo, [1]). When $m = 1$, instead, it is well known that it can be solved in polynomial time (see, for instance, Bodin et al. [3] and Carraresi and Gallo [7]). Note that when costs $\bar{\gamma}_{k,j}$ and $\tilde{\gamma}_{j,k}$ are independent of the depot D_k, as for objective a) above, the problem reduces to the single depot case and hence it is polynomially solvable. Also note that the problem of finding (if any) a feasible solution to MD-VSP, i.e., a solution to the problem defined by i), ii), and iii) above, reduces to the single depot case by disregarding all finite cost differences, and hence is polynomially solvable.

In the following we will assume that a feasible solution always exists, i.e. that a sufficient number of vehicles is available.

Heuristic algorithms have been proposed for *MD-VSP* by Bodin, Rosenfield and Kydes [4], Smith and Wren [10], and Bertossi, Carraresi, and Gallo [1]. Surveys on the subject can be found in Bodin and Golden [2], Wren [11], Bodin et al. [3], and Carraresi and Gallo [6,7]. To our knowledge, no optimal algorithm has been proposed.

In this paper we present a branch and bound algorithm for the optimal solution of *MD-VSP*. In Section II, graph theory and integer linear programming models are given. Different lower bounds are introduced in Section III, and combined to obtain an additive lower bounding procedure in Section IV. Section V describes dominance criteria allowing fathomings of large subsets of solutions. A branch and bound algorithm based on these lower bounds and dominance criteria is designed in Section VI and computationally analyzed in Section VII.

II. MODELS

We first introduce a graph theory model for *MD-VSP*. Consider a digraph $G = (V,A)$ defined as follows. $V = \{1, \ldots, p\}$ (where $p = n + \sum_{k=1}^{m} r_k$) is the vertex set, partitioned into $m + 1$ subsets: subset $N = \{1, \ldots, n\}$, in which vertex j is associated with trip T_j, and subsets $W_1, \ldots, W_m$, where, for $k = 1, \ldots, m$, $W_k = \{n + \sum_{h=1}^{k-1} r_h + 1, \ldots, n + \sum_{h=1}^{k} r_h\}$ contains r_k vertices associated with the r_k vehicles stationed at depot D_k. $A = \{(i,j): i,j \in V\}$ is the arc set. For each arc $(i,j) \in A$, we define the corresponding cost

$$c_{i,j} = \begin{cases} \gamma_{i,j}, & \text{if } i,j \in N; \\ \bar{\gamma}_{k,j}, & \text{if } i \in W_k, j \in N; \\ \tilde{\gamma}_{i,k}, & \text{if } i \in N, j \in W_k; \\ 0, & \text{if } i,j \notin N, i = j; \\ \infty, & \text{if } i,j \notin N, i \neq j. \end{cases}$$

MD-VSP can be stated as the problem of finding a set of circuits (*subtours*), each containing only one vertex in $V \setminus N$, such that all vertices in V are visited exactly once and the sum of the costs of the arcs in the solution is minimized. Each subtour having finite cost and visiting sequentially vertices $v_1, v_2, \ldots, v_h, v_1$, with $v_1 \in W_k$ (for some k) and $v_2, \ldots, v_h$ in N, corresponds to the duty of a vehicle stationed at depot D_k and covering trips $T_{v_2}, \ldots, T_{v_h}$, and its cost is the cost of this duty. The subtours containing only one vertex v_1 (with $v_1 \notin N$) have zero cost and correspond to unused vehicles. Conversely, it is easy to see that any feasible duty of a vehicle corresponds to a feasible subtour having the same cost. Therefore the global cost of the subtours in the optimal solution is equal to the overall cost of *MD-VSP* (if such a global cost is infinite, no feasible solution for *MD-VSP* exists).

Note that no finite cost subtour exists in the subgraph induced by vertex set N, so any subtour must visit at least one vertex in $V \setminus N$. We require no more than one vertex in $V \setminus N$ to be visited by each feasible subtour, thus avoiding vehicles starting and finishing their duty at different depots.

The graph theory model can be formulated as an integer linear programming problem as follows. Let $x_{i,j} = 1$ if arc $(i,j) \in A$ is used in the optimal solution, $= 0$ otherwise, and define Π as the family of all elementary paths $\mathcal{P}$ joining different vertices in $V \setminus N$.

$$(MD\text{-}VSP) \qquad v(MD\text{-}VSP) = \min \sum_{i \in V} \sum_{j \in V} c_{i,j} x_{i,j} \tag{1}$$

subject to

$$\sum_{i \in V} x_{i,j} = 1, \qquad j \in V \tag{2}$$

$$\sum_{j \in V} x_{i,j} = 1, \qquad i \in V \tag{3}$$

$$\sum_{(i,j) \in \mathcal{P}} x_{i,j} \leq |\mathcal{P}| - 1, \qquad \mathcal{P} \in \Pi \tag{4}$$

$$x_{i,j} \in \{0,1\}, \qquad i,j \in V. \tag{5}$$

Constraints (2) and (3), imposing that each vertex is visited exactly once, ensure that each feasible solution corresponds to a family of disjoint subtours covering all the vertices. Constraints (4) ensure that each subtour visits only one vertex in $V \setminus N$, since no path $\mathcal{P}$ joining different vertices in $V \setminus N$ is allowed.

It is worth noting that family Π of the forbidden paths can be restricted to contain only elementary paths joining vertices v_1 and v_2 with $v_1 \in W_{k_1}$, $v_2 \in W_{k_2}$ and $k_1 \neq k_2$, that is, paths joining vertices corresponding to vehicles stationed at different depots. In this way, subtours visiting different vertices v_1 and v_2 with $v_1, v_2 \in W_k$ are allowed. However, it is always possible to obtain an equivalent solution in which the subtour visiting vertex v_1 does not visit vertex v_2. In fact, let $(v_1,i_1, \ldots ,i_h,v_2,j_1, \ldots ,j_l,v_1)$ be the sequence of vertices visited in the subtour. Such a subtour can be split, with no extra cost, into two subtours $(v_1,i_1, \ldots ,i_h,v_1)$ and $(v_2,j_1, \ldots ,j_l,v_2)$. This last definition of family Π will be used throughout the paper.

As for the single depot case, family Π is empty, so *MD-VSP*, defined by (1) to (5), reduces to the well-known *min-sum linear assignment problem* (AP), defined by (1), (2), (3), and (5), which is solvable in $O(|V|^3)$ (i.e., $O(n^3)$) time.

III. LOWER BOUNDS

A first lower bound, $\delta^{(0)}$, for *MD-VSP* can be obtained by removing constraints (4) and optimally solving the corresponding assignment problem *AP* ($\delta^{(0)}$ = optimum value of *AP*). Let $(\bar{x}_{i,j})$ be an optimal solution to *AP*, defining a family of subtours covering all the vertices of graph G. Since constraints (4) have not been taken into account, a subtour could visit vertices associated with different depots thus leading to an infeasible solution in which a vehicle does not return to its depot.

As for a different lower bound, consider a vertex $h \in N$, associated with trip T_h, and suppose it is covered by a vehicle stationed, say, at depot D_q. In this hypothesis, we may require the existence of two (vertex disjoint) paths $\mathcal{P}_1$ and $\mathcal{P}_2$ in the optimal solution to *MD-VSP*, where path $\mathcal{P}_1$ joins a vertex in W_q to vertex h, path $\mathcal{P}_2$ joins vertex h to a vertex in W_q, and neither $\mathcal{P}_1$ nor $\mathcal{P}_2$ visit any vertex in $Q = V \setminus (N \cup W_q)$ (i.e. any vertex associated with a depot different from D_q). Let $\lambda_{h,q}$ (resp. $\bar{\lambda}_{h,q}$) be the cost of a minimum-cost path $\mathcal{P}_1$ (resp. $\mathcal{P}_2$) satisfying the above conditions. Values $\lambda_{h,q}$ and $\bar{\lambda}_{h,q}$ can be computed through any shortest-path algorithm on the subgraph of G induced by vertex set $V \setminus Q$. A valid lower bound $\delta_{h,q}$ for *MD-VSP*, when restricted to having trip T_h covered by a vehicle stationed at depot D_q, is then

$$\delta_{h,q} = \lambda_{h,q} + \bar{\lambda}_{h,q}.$$

As for the unrestricted *MD-VSP*, a valid lower bound $\delta^{(h)}$ (for $h = 1, \ldots ,n$) is given by

$$\delta^{(h)} = \min\{\delta_{h,q}: q = 1, \ldots ,m\},$$

since trip T_h must be covered by a vehicle stationed at some depot.

Since none of the proposed bounds dominates the others, a possible way to obtain a strengthened bound is to compute the maximum among $\delta^{(0)}$ and $\delta^{(1)}, \ldots ,\delta^{(n)}$.

Note that bound $\delta^{(0)}$ exploits the condition that each trip must be covered exactly once, but cannot ensure that each vehicle starts and finishes its duty at the same depot. On the other hand, each bound $\delta^{(h)}$ ($h = 1, \ldots ,n$) takes into account the condition that the vehicle covering trip T_h must start and finish its duty at the same depot, but cannot ensure the covering of all the trips, since only one vehicle is considered. To exploit the complementariness of the proposed bounds, we use a lower bounding procedure based on the *additive approach* proposed by Fischetti and Toth [8].

IV. AN ADDITIVE LOWER BOUNDING PROCEDURE

Let $\mathcal{L}^{(0)}$ be the lower bounding procedure computing value $\delta^{(0)}$, and $\mathcal{L}^{(h)}$ that computing value $\delta^{(h)}$ ($h = 1, \ldots ,n$). Suppose that, for $h = 0,1, \ldots ,n$ and for any cost matrix $\bar{c} = (\bar{c}_{i,j})$, procedure $\mathcal{L}^{(h)}(\bar{c})$, when applied to the instance of *MD-VSP* having cost matrix $\bar{c}$, returns its lower bound $\delta^{(h)}$ as well as a *residual cost* matrix $c^{(h)}$ such that:

i) $c_{i,j}^{(h)} \geq 0$ for each $i,j \in V$;

ii) $\delta^{(h)} + \Sigma_{i\in V} \Sigma_{j\in V} c_{i,j}^{(h)} x_{i,j} \leq \Sigma_{i\in V} \Sigma_{j\in V} \bar{c}_{i,j} x_{i,j}$, for each feasible solution $(x_{i,j})$ for *MD-VSP*.

The additive approach generates a sequence of instances of problem *MD-VSP*, each obtained by considering the residual cost matrix corresponding to the previous instance and applying a different bounding procedure. A Pascal-like outline of the approach follows.

Algorithm ADD-MDVSP:

1. input: cost matrix c;
2. output: lower bound δ, final residual-cost matrix $c^{(n)}$;
begin
3. initialize $c^{(-1)} := c$, $\delta := 0$;
4. **for** $h := 0$ **to** n **do**
begin
5. apply $\mathcal{L}^{(h)}(c^{(h-1)})$, thus obtaining value $\delta^{(h)}$ and the residual-cost matrix $c^{(h)}$;
6. $\delta := \delta + \delta^{(h)}$
end
end.

In order to show that each value δ computed at step 6 is a valid lower bound for *MD-VSP*, consider step 5 at iteration h ($h = 0,1, \ldots ,n$), and the corresponding instance of *MD-VSP* defined by cost matrix $c^{(h-1)}$ (the current value of δ is $\Sigma_{l=0}^{h-1} \delta^{(l)}$). The following problem:

$$\delta + \min \sum_{i\in V} \sum_{j\in V} c_{i,j}^{(h-1)} x_{i,j}$$

subject to (2), (3), (4) and (5)

turns out to be a relaxation of the original *MD-VSP*. In fact, we show by induction on h that, for all feasible $(x_{i,j})$,

$$\sum_{l=0}^{h-1} \delta^{(l)} + \sum_{i\in V} \sum_{j\in V} c_{i,j}^{(h-1)} x_{i,j} \leq \sum_{i\in V} \sum_{j\in V} c_{i,j} x_{i,j} \tag{6}$$

holds for $h = 0,1, \ldots ,n$. Case $h = 0$ is trivially true. Suppose now that inequality (6) holds for $h = \bar{h}$. From condition ii) on the residual-cost matrix, we have

$$\sum_{l=0}^{\bar{h}-1} \delta^{(l)} + \delta^{(\bar{h})} + \sum_{i\in V} \sum_{j\in V} c_{i,j}^{(\bar{h})} x_{i,j} \leq \sum_{l=0}^{\bar{h}-1} \delta^{(l)} + \sum_{i\in V} \sum_{j\in V} c_{i,j}^{(\bar{h}-1)} x_{i,j},$$

so inequality (6) holds for $h = \bar{h} + 1$ as well.

Value δ computed at step 6 is then a valid lower bound for *MD-VSP*, since, because of condition i) on the residual costs, $\Sigma_{i \in V} \Sigma_{j \in V} c_{i,j}^{(h-1)} x_{i,j} \geq 0$ for each $(x_{i,j})$ feasible solution for *MD-VSP*.

The sequence of values δ is nondecreasing, since increments $\delta^{(h)}$ are clearly nonnegative for $h = 1, \ldots, n$.

A. Computation of the Residual Costs

The key step of algorithm *ADD-MDVSP* is the computation of valid residual-cost matrices corresponding to the proposed lower bounds.

As for bounding procedure $\mathcal{L}^{(0)}(c)$, let $(u_i^*) - (v_j^*)$ be the optimal solution to the linear programming dual problem associated with *AP*. The reduced costs $c_{i,j}^* = c_{i,j} - u_i^* - v_j^*$ are then valid residual costs corresponding to lower bound $\delta^{(0)}$ = optimum value of $AP = \Sigma_{i=1}^{n} (u_i^* + v_i^*)$, since both conditions i) and ii) are satisfied.

As for bounding procedure $\mathcal{L}^{(h)}(c)$, for $h = 1, \ldots, n$, we proceed as follows. First, we consider the problem, *SPP*, of finding, in graph G, a minimum cost path from a given vertex s to a different vertex t without visiting the vertices of a given set $Q \subset V$. We describe how to solve this problem and compute the corresponding reduced costs, which are then used to determine valid residual costs associated with values $\delta_{h,q}$, with $q = 1, \ldots, m$. Finally, we show how to compute the overall residual costs corresponding to lower bound $\delta^{(h)} = \min\{\delta_{h,q}: q = 1, \ldots, m\}$.

Computation of the Reduced Costs Associated with SPP

Problem *SPP* can be formulated through the following linear programming model:

$$(SPP) \qquad v(SPP) = \min \sum_{i \in V} \sum_{j \in V} c_{i,j} \, y_{i,j}$$

subject to

$$\sum_{i \in V \setminus Q} y_{i,v} - \sum_{j \in V \setminus Q} y_{v,j} = \begin{cases} -1, & \text{for } v = s, \\ 1, & \text{for } v = t, \\ 0, & \text{for } v \in V \setminus (Q \cup \{s,t\}); \end{cases}$$

$$y_{i,j} = 0, \qquad i,j \in V: i \in Q \text{ or } j \in Q;$$

$$y_{i,j} \geq 0, \qquad i,j \in V;$$

where $y_{i,j} = 1$ iff vertex j is visited just after vertex i in the optimal path.

Problem *SPP* can be solved by removing from graph G all vertices in Q and by applying a shortest path algorithm to determine cost $\bar{L}_i$ of the shortest path from vertex s to vertex i, for each $i \in V \setminus Q$. Hence we have $v(SPP) = \bar{L}_t$. In order to compute the corresponding reduced costs $c^*_{i,j}$, let us consider the dual problem associated with SPP:

$$(D - SPP) \qquad v(D - SPP) = \max(L_t - L_s)$$

subject to

$$c_{i,j} + L_i - L_j \geq 0, \qquad i,j \in V \setminus Q;$$

$$c_{i,j} + \sigma_{i,j} \geq 0, \qquad i,j \in V: i \in Q \text{ or } j \in Q.$$

An optimal solution for this problem is determined by $L_i = \bar{L}_i$ for $i \in V \setminus Q$, and $\sigma_{i,j} = \infty$ for $i,j \in V$: $i \in Q$ or $j \in Q$. In fact, the dual feasibility conditions $c_{i,j} + \bar{L}_i - \bar{L}_j \geq 0$ (for $i,j \in V \setminus Q$) are clearly satisfied because of the definition of the shortest path values $\bar{L}_i$'s. As for the optimality, it is enough to note that the value of the dual solution $\bar{L}_t - \bar{L}_s$, is equal to the optimal value $\bar{L}_t$ of the primal problem. The corresponding reduced costs are $\bar{c}_{i,j} = c_{i,j} + \bar{L}_i - \bar{L}_j$ for $i,j \in V \setminus Q$, $\bar{c}_{i,j} = \infty$ for $i,j \in V$: $i \in Q$ or $j \in Q$.

Alternative reduced costs can be obtained by considering the equivalent optimal dual solution given by $L_i = L^*_i$, for $i \in V \setminus Q$, and $\sigma_{i,j} = \infty$, for $i,j \in V$: $i \in Q$ or $j \in Q$, where

$$L^*_i = \min\{\bar{L}_i, \bar{L}_t\}, \qquad i \in V \setminus Q.$$

This dual solution has the same value $L^*_t = \bar{L}_t$ as the previous one. As for dual feasibility, note that the reduced cost $c^*_{i,j}$ defined by

$$c^*_{i,j} = \begin{cases} c_{i,j} + L^*_i - L^*_j, & \text{if } i,j \in V \setminus Q \\ \infty, & \text{otherwise} \end{cases}$$

can be greater, equal or less than the previous reduced cost $\bar{c}_{i,j}$. In particular, we can show that:

a) $\bar{c}_{i,j} \leq c^*_{i,j} \leq c_{ij}$ if $\bar{c}_{i,j} < c_{i,j}$: in this case, in fact, $\bar{L}_i < \bar{L}_j$ and then $0 \leq L^*_i - \bar{L}^*_j \leq \bar{L}_j - \bar{L}_i$;

b) $c_{ij} \leq c^*_{i,j} \leq \bar{c}_{i,j}$ if $\bar{c}_{i,j} > c_{i,j}$: in this case, in fact, either $\bar{c}_{i,j} = c^*_{i,j} = \infty$, or $\bar{L}_i > \bar{L}_j$ and then $0 \leq L^*_i - L^*_j \leq \bar{L}_i - \bar{L}_j$;

c) $c_{i,j} = c^*_{i,j} = \bar{c}_{i,j}$ if $\bar{c}_{i,j} = c_{i,j}$: in this case, in fact, $\bar{L}_i = \bar{L}_j$ and then $L^*_i = L^*_j$.

Hence, $c^*_{i,j} \geq \min\{c_{i,j}, \bar{c}_{i,j}\} \geq 0$ for all $i,j \in V$. So the alternative solution is dual-feasible.

In the following we will consider the $c^*_{i,j}$'s as the reduced costs corresponding to SPP since, as we will see, they turn out to be more effective than the $\bar{c}_{i,j}$'s for our approach.

It is worth noting that for each feasible solution $(y_{i,j})$ for *SPP* we have

$$
\begin{aligned}
& v(SPP) + \sum_{i\in V}\sum_{j\in V} c^*_{i,j}\, y_{i,j} \\
&= L^*_t - L^*_s + \sum_{i\in V}\sum_{j\in V} c_{i,j}\, y_{i,j} + \sum_{i\in V\setminus Q}\sum_{j\in V\setminus Q} (L^*_i - L^*_j)\, y_{i,j} \\
&= L^*_t - L^*_s + \sum_{i\in V}\sum_{j\in V} c_{i,j}\, y_{i,j} + \sum_{v\in V\setminus Q} L^*_v \left(\sum_{j\in V\setminus Q} y_{v,j} - \sum_{i\in V\setminus Q} y_{i,v}\right),
\end{aligned}
$$

and hence:

$$
v(SPP) + \sum_{i\in V}\sum_{j\in V} c^*_{i,j}\, y_{i,j} = \sum_{i\in V}\sum_{j\in V} c_{i,j}\, y_{i,j}. \tag{7}
$$

Computation of the Residual Costs Associated with $\delta_{h,q}$

Consider first the computation of the value $\delta_{h,q} = \lambda_{h,q} + \bar{\lambda}_{h1,q}$, for a given q. Introduce an extended graph $G' = (V',A')$, with $V' = V \cup \{0\}$, and $A' = A \cup \{(0, j): j \in W_q\} \cup \{(j,0): j \in W_q\}$, and define $c_{i,j} = 0$ for the new arcs $(i,j) \in A'\setminus A$.

Value $\bar{\lambda}_{h,q}$ can be computed by solving an instance of SPP defined by graph G', $s = 0, t = h, Q = V\setminus(N \cup W_q)$. Let $c^{(h,q)}$ be the corresponding reduced-cost matrix.

Value $\bar{\lambda}_{h,q}$, and the corresponding reduced-cost matrix $\bar{c}^{(h,q)}$, can be similarly computed by considering graph G', $s = h$, $t = 0$, $Q = V\setminus(N \cup W_q)$.

Let us define:

$$
\gamma^{(h,q)}_{i,j} = \min\{c^{(h,q)}_{i,j}, \bar{c}^{(h,q)}_{i,j}, c_{i,j}\} \qquad \text{for all } i,j \in V.
$$

The $\gamma^{(h,q)}_{i,j}$'s can be considered "residual costs" associated with $\delta_{h,q}$, in the sense that the following proposition holds:

Proposition 1. For each feasible solution $(x_{i,j})$ for MD $-$ VSP in which trip T_h is covered by a vehicle stationed at depot D_q, we have:

$$
\delta_{h,q} + \sum_{i\in V}\sum_{j\in V} \gamma^{(h,q)}_{i,j}\, x_{i,j} \leq \sum_{i\in V}\sum_{j\in V} c_{i,j}\, x_{i,j}.
$$

Proof. Any feasible solution $(x_{i,j})$ can be "decomposed" into three "partial" solutions $(y^{(1)}_{i,j})$, $(y^{(2)}_{i,j})$ and $(y^{(3)}_{i,j})$ such that

From the definition of the reduced (and residual) cost matrices $c^{(h,q)}$ and $\bar{c}^{(h,q)}$, and from (7), it follows that

$$
\lambda_{h,q} + \sum_{i\in V}\sum_{j\in V} c^{(h,q)}_{i,j}\, y^{(1)}_{i,j} = \sum_{i\in V}\sum_{j\in V} c_{i,j}\, y^{(1)}_{i,j},
$$

and

$$\bar{\lambda}_{h,q} + \sum_{i \in V} \sum_{j \in V} \bar{c}_{i,j}^{(h,q)} y_{i,j}^{(2)} = \sum_{i \in V} \sum_{j \in V} c_{i,j} y_{i,j}^{(2)}.$$

Hence

$$\alpha \equiv (\lambda_{h,q} + \bar{\lambda}_{h,q} + 0) + \sum_{i \in V} \sum_{j \in V} (c_{i,j}^{(h,q)} y_{i,j}^{(1)} + \bar{c}_{i,j}^{(h,q)} y_{i,j}^{(2)} + c_{i,j} y_{i,j}^{(3)})$$
$$= \sum_{i \in V} \sum_{j \in V} c_{i,j} (y_{i,j}^{(1)} + y_{i,j}^{(2)} + y_{i,j}^{(3)}) = \sum_{i \in V} \sum_{j \in V} c_{i,j} x_{i,j}.$$

i) $x_{i,j} = y_{i,j}^{(1)} + y_{i,j}^{(2)} + y_{i,j}^{(3)}$ for all $i,j \in V$;
ii) $(y_{i,j}^{(1)})$ is a feasible solution for problem SPP with $s \in W_q$, $t = h$, and $Q = V \setminus (N \cup W_q)$;
iii) $(y_{i,j}^{(2)})$ is a feasible solution for problem SPP with $s = h$, $t \in W_q$, and $Q = V \setminus (N \cup W_q)$;
iv) $y_{i,j}^{(3)} \geq 0$ for all $i,j \in V$.
So, from the definition of the $\gamma_{i,j}^{(h,q)}$'s,

$$\delta_{h,q} + \sum_{i \in V} \sum_{j \in V} \gamma_{i,j}^{(h,q)} (y_{i,j}^{(1)} + y_{i,j}^{(2)} + y_{i,j}^{(3)}) \leq \alpha = \sum_{i \in V} \sum_{j \in V} c_{i,j} x_{i,j}.$$ ■

Note that, because of its definition, no residual cost $\gamma_{i,j}^{(h,q)}$ can exceed the corresponding original cost $c_{i,j}$. We can now justify our claim that, for SPP, the reduced costs $c_{i,j}^*$'s we use are more effective than the $\bar{c}_{i,j}$'s: each value $\gamma_{i,j}^{(h,q)}$ computed through the reduced costs $c_{i,j}^*$ is, in fact, consistently not less than the value we would obtain through the reduced cost $\bar{c}_{i,j}$.

Computation of the Residual Costs Associated with Bound $\delta^{(h)}$

We now show that valid overall residual costs $\gamma_{i,j}^{(h)}$ corresponding to $\delta^{(h)} = \min\{\delta_{h,q}: q = 1, \ldots, m\}$ can be computed as:

$$\gamma_{i,j}^{(h)} = \min\{\gamma_{i,j}^{(h,q)}: q = 1, \ldots, m\} \qquad \text{for all } i,j \in V.$$

In fact:

Proposition 2. For each feasible solution (x_{ij}) for MD − VSP we have:

$$\delta^{(h)} + \sum_{i \in V} \sum_{j \in V} \gamma_{i,j}^{(h)} x_{i,j} \leq \sum_{i \in V} \sum_{j \in V} c_{i,j} x_{i,j}.$$

Proof. Let $(\tilde{x}_{i,j})$ be any feasible solution for MD − VSP in which trip T_h is covered by a vehicle stationed, say, at depot $\tilde{q}$. From Proposition 1 we have:

$$\delta^{(h)} + \sum_{i \in V} \sum_{j \in V} \gamma_{i,j}^{(h)} \tilde{x}_{i,j} \leq \delta_{h,\tilde{q}} + \sum_{i \in V} \sum_{j \in V} \gamma_{i,j}^{(h,\tilde{q})} \tilde{x}_{i,j} \leq \sum_{i \in V} \sum_{j \in V} c_{i,j} \tilde{x}_{i,j}.$$

The thesis follows from the arbitrariness of solution $(\tilde{x}_{i,j})$. ■

B. An Efficient Implementation of ADD–MDVSP

An efficient implementation of algorithm ADD–MDVSP is given below.

1. solve the assignment problem AP on the original cost matrix $(c_{i,j})$ and let $(\bar{c}_{i,j})$ be the corresponding reduced-cost matrix;
   $\delta := $ optimum value of *AP;*
   repeat
2.   **comment** find values $\delta_{h,q} = \lambda_{h,q} + \bar{\lambda}_{h,q}$ for each trip T_h and for each depot D_q;
3. **for** $q := 1$ **to** m **do**
      begin
4.   compute costs $\lambda_{h,q}$ of the "forward" minimum-cost paths from a vertex in W_q to all vertices $h \in N \cup W_q$, not visiting vertices in $V \setminus (N \cup W_q)$;
5.   compute costs $\bar{\lambda}_{h,q}$ of the "backward" minimum-cost paths from all vertices $h \in N \cup W_q$ to a vertex in W_q, not visiting vertices in $V \setminus (N \cup W_q)$
      end;
6. **comment** find values $\delta^{(h)}$ for each trip T_h;
7. **for each** $h \in N$ **do** $\delta^{(h)} := \min\{\lambda_{h,q} + \bar{\lambda}_{h,q}: q = 1, \ldots, m\}$;
8. **find trips** T_{h_1} **and** T_{h_2} **having the two largest values of** $\delta^{(h)}$, with $\delta^{(h_1)} \geqslant \delta^{(h_2)}$;
9. **if** $\delta^{(h_1)} > 0$ **then**
      begin
10.   **comment** increase the lower bound δ;
11. $\delta := \delta + \delta^{(h_1)}$;
12.   **comment** decrease labels $\lambda_{h,q}$ and $\bar{\lambda}_{h,q}$ to obtain more effective residual costs;
13.   **for** $q := 1$ **to** m **do**
14.   **for each** $h \in N \cup W_q$ **do**
15. $\lambda_{h,q} := \min\{\lambda_{h,q}, \lambda_{h_1,q}\}$, $\bar{\lambda}_{h,q} := \min\{\bar{\lambda}_{h,q}, \bar{\lambda}_{h_1}\}$;
16.   **comment** update the residual cost matrix $(\bar{c}_{i,j})$;
17.   **for** $q := 1$ **to** m **do**
18.   **for each** arc (i,j): $i,j \in N \cup W_q$ **do**
19. $\bar{c}_{i,j} := \min\{\bar{c}_{i,j} + \lambda_{i,q} - \lambda_{j,q}, \bar{c}_{i,j} - \bar{\lambda}_{i,q} + \bar{\lambda}_{j,q}, \bar{c}_{i,j}\}$
       end
20. **until** $\delta^{(h_2)} = 0$.

At step 4, the forward shortest path costs $\lambda_{h,q}$ are computed through any $O(n^2)$ shortest path algorithm (possibly specialized for acyclic graphs) on the current residual cost matrix $(\bar{c}_{i,j})$ and on the vertex set $N \cup W_q$. At step 5, the backward shortest path costs $\bar{\lambda}_{h,q}$ are computed in a similar way, (implicitly) considering the transposed matrix $(\bar{c}_{i,j})^T$. It is worth noting that, for the forward paths, the corresponding reduced costs are $\bar{c}_{i,j} + \lambda_{i,q} - \lambda_{j,q}$, while, for the backward ones, they are $c_{i,j} - \bar{\lambda}_{i,q} + \bar{\lambda}_{j,q}$.

As for time complexity, step 1 requires $O(|V|^3)$ time, while steps 3 to 5 and steps 17 to 19 require $O(mn^2)$ time and can be executed at most n times.

The implementation differs from the previously described additive algorithm mainly because, at each iteration of the repeat-until loop, the best lower bound, $\delta^{(h_1)}$, is used to increase the current bound δ. This allows the algorithm to be stopped as soon as no further improvement can arise (i.e., when $\delta^{(h_2)} = 0$), while the overall time complexity is preserved.

V. A DOMINANCE CRITERION

We now describe a dominance criterion based on the general approach introduced by Fischetti and Toth [9].

Let us suppose solving MD–VSP through a branch and bound algorithm, and consider the current node α of the branch decision tree. Define the corresponding *partial solution* through arc sets $I^{(\alpha)}$ and $F^{(\alpha)}$ containing, respectively, the arcs "imposed" and "forbidden" at the previous levels. In the branch and bound algorithm we propose (see Section VI), the branching rule, based on subtour elimination, is such that set $I^{(\alpha)}$ can be partitioned into q paths $\mathcal{P}_1^{(\alpha)}, \mathcal{P}_2^{(\alpha)}, \ldots, \mathcal{P}_q^{(\alpha)}$, the hth starting from a vertex in W_{k_h} and finishing at vertex $j_h \in N$. We can now define the *state* associated with node α as $(S, q, (k_1, \ldots, k_q), (j_1, \ldots, j_q), c, F^{(\alpha)})$, where S = {vertices visited by paths $\mathcal{P}_1^{(\alpha)}, \ldots, \mathcal{P}_q^{(\alpha)}$}, and c is the sum of the arc costs in the q paths. The existence of a node β having a state $(S, q, (k_1, \ldots, k_q), (j_1, \ldots, j_q), c', F^{(\beta)})$ such that $c' < c$ and $F^{(\beta)} \subseteq F^{(\alpha)}$, is a sufficient condition to fathom node α. A way to apply this dominance criterion might be to store the state of all the nodes generated so far and compare them with the state of the current node α. This method, however, is impractical, even for small instances, because of the memory requirement.

Consider, instead, the following alternative approach. Let us define the *auxillary problem* associated with node α:

$$(\mathrm{XP}^{(\alpha)}) \qquad v(\mathrm{XP}^{(\alpha)}) = \min \sum_{i \in S} \sum_{j \in S} c_{i,j}\, x_{i,j}$$

subject to

$$\left\{ \begin{array}{l} (x_{i,j}\colon i,j \in S) \text{ defines } q \text{ paths, visiting all} \\ \text{the vertices in } S \text{ once, the } h\text{th path starting from} \\ \text{a vertex in } W_{k_h} \text{ and finishing at vertex } j_h \end{array} \right\}.$$

Let $(x^*_{i,j}\colon i,j \in S)$ be an optimal solution to $\mathrm{XP}^{(\alpha)}$ defining a partial solution of MD–VSP associated with a node β^*, with $I^{(\beta^*)} = \{(i,j)\colon\ x^*_{i,j} = 1,\ i,j \in S\}$ and $F^{(\beta^*)} = \emptyset$. The state associated with node β^* is clearly $(S, q, (k_1, \ldots, k_q), (j_1, \ldots, j_q), v(\mathrm{XP}^{(\alpha)}), \emptyset)$. So $v(\mathrm{XP}^{(\alpha)}) < c$ is a sufficient condition to fathom node α.

The auxiliary problem $\mathrm{XP}^{(\alpha)}$ is as difficult as MD–VSP, although with a smaller size. In fact, any instance of MD–VSP can easily be transformed into an equivalent instance of $\mathrm{XP}^{(\alpha)}$ as follows. Introduce, for each vertex $i \in V \setminus N$, a new vertex σ_i whose incident arcs have the same costs as those incident at i, and set $c_{i,\sigma_i} = 0$. Now define

$$S = V \cup \{\sigma_i : i \in V \setminus N\},$$

$$q = r_1 + r_2 + \cdots + r_m = |V \setminus N|,$$

$$(k_1, \ldots, k_q) = (\underbrace{1, \ldots, 1}_{r_1 \text{ times}}, \underbrace{2, \ldots, 2}_{r_2 \text{ times}}, \ldots, \underbrace{m, \ldots, m}_{r_m \text{ times}}),$$

$$(j_1, \ldots, j_q) = (\{\sigma_i : i \in W_1\}, \{\sigma_i : i \in W_2\}, \ldots, \{\sigma_i : i \in W_m\}).$$

It is easy to verify that any optimal solution of this instance of $XP^{(\alpha)}$ determines an optimal solution of the instance MD–VSP as well.

In our approach the auxiliary problem is solved through a heuristic algorithm producing a feasible solution of value $h(XP^{(\alpha)})$. This leads to the valid, but generally weaker, dominance criterion: *fathom node* α *if* $h(XP^{(\alpha)}) < c$.

The heuristic algorithm we use is based on the consideration that $XP^{(\alpha)}$ can be solved in polynomial time when all its paths start from the same depot. So we can choose a depot D_f, with $f \in \{k_1, k_2, \ldots, k_q\}$, and "fix" $x^*_{i,j} = 1$ for each arc (i,j) belonging to a path $\mathcal{P}_g^{(\alpha)}$ with $g \in \{1, \ldots, q\}$ and such that $k_g \neq f$, i.e., to a path imposed in the partial solution of MD–VSP corresponding to node α and not starting from a vertex in W_f. Problem $XP^{(\alpha)}$ reduces now to an easier one, in which all "free" vertices (i.e., all vertices not incident at "fixed" arcs) must be covered by vehicles stationed at depot D_f. Let XP′ be such a reduced problem, defined by $S' = \{$all free vertices in $S\}$, q' = number of paths $\mathcal{P}_i^{(\alpha)}$'s starting from a vertex in W_f, $(k'_1, \ldots, k'_{q'}) = (4f, \ldots, f)$, $(j'_1, \ldots, j'_{q'})$ = (final vertices of paths $\mathcal{P}_i^{(\alpha)}$'s starting from a vertex in W_f). Problem XP′ can be optimally solved in $O(|S'|^3)$ time, since it is equivalent to an instance of MD–VSP with a single depot, D_f, in which q' vehicles are stationed, and with the cost of each arc leaving one of the vertices $j'_1, \ldots, j'_{q'}$ changed to 0 if the arc enters a vertex in W_f, to ∞ otherwise. The overall value $h(XP^{(\alpha)})$ is then $v(XP')$ + (sum of the arc costs of the paths $\mathcal{P}_i^{(\alpha)}$'s not starting from a vertex in W_f).

The above dominance criterion can be strengthened by introducing a ranking among equivalent feasible solutions. This is obtained by replacing, in the definition of MD–VSP, the original costs $(c_{i,j})$ with $(c_{i,j} + \varepsilon_{i,j})$, where the $\varepsilon_{i,j}$'s are values small enough to preserve the original ranking among the feasible solutions of MD–VSP, and able to differentiate possible equivalent solutions. In this way our dominance criterion can be more successful in dealing with situations in which, considering the original costs, the heuristic algorithm would find a feasible partial solution different from the current one, but having the same cost (i.e., when $h(XP^{(\alpha)}) = c$).

VI. A BRANCH AND BOUND ALGORITHM

We now describe a branch and bound algorithm for the optimal solution of MD–VSP, based on the additive lower bound and the dominance criterion proposed in the previous sections.

As for the branching rule, we use an adaptation of the subtour elimination scheme for the Asymmetric Travelling Salesman Problem, as implemented by Carpaneto and Toth [5]. At each node α of the branch decision tree we consider the solution of the corresponding assignment problem, AP, found by step 1 of the bounding procedure

ADD–MDVSP of Section IV.B. If such a solution is feasible for MD–VSP, we update the incumbent optimal solution. Otherwise, we choose a subtour, Φ, visiting vertices associated with different depots and a path $\mathcal{P}^{(\alpha)}$ in Φ starting from a vertex in W_{k_1} and finishing at a vertex in W_{k_2} (with $k_1 \neq k_2$), thus violating constraints (4). Let $\mathcal{P}^{(\alpha)} = \{(v_1,v_2), (v_2,v_3), \ldots, (v_h, v_{h+1})\}$ (with $v_1 \in W_{k_1}$, $v_{h+1} \in W_{k_2}$, and $v_2, v_3, \ldots, v_h \in N$). The current node α generates h descendent nodes $\alpha_1, \alpha_2, \ldots, \alpha_h$, whose sets $I^{(\alpha_i)}$ and $F^{(\alpha_i)}$, containing the imposed and the forbidden arcs, are defined by:

$$\left.\begin{aligned} I^{(\alpha_i)} &= I^{(\alpha)} \cup \{(v_1,v_2), \ldots, (v_{i-1},v_i)\} \\ F^{(\alpha_i)} &= F^{(\alpha)} \cup \{(v_i,v_{i+1})\} \end{aligned}\right\} i = 1, \ldots, h.$$

Nodes α_i for which $I^{(\alpha_i)} \cap F^{(\alpha_i)} \neq \emptyset$ are clearly not generated.

For each descendent node α_i, we apply our additive lower bounding procedure, and fathom the node as soon as the current lower bound δ becomes greater or equal to the value of the incumbent solution. If the node is not fathomed at the end of the bounding procedure, we apply the dominance criterion. If node α_i remains unfathomed, we choose an infeasible path $\mathcal{P}^{(\alpha_i)}$ in the optimal solution of AP associated with α_i, and stored both node α_i and path $\mathcal{P}^{(\alpha_i)}$ in the queue of the active nodes of the branch decision tree.

As for the heuristic procedure applied at node α_i to check the dominance criterion, we choose D_f as depot D_{k_1}, that is, as the depot from which the infeasible path $\mathcal{P}^{(\alpha)}$ to be broken starts. In fact, it is useless to consider a different depot, since the "imposed" paths starting from it have not been augmented at node α_i with respect to the parent node α. For the same reason, no dominance criterion is applied for node α_1. In addition, note that all nodes $\alpha_{i+1}, \ldots, \alpha_h$ can be immediately fathomed when node α_i is fathomed through the dominance criterion.

To strengthen the dominance criterion, each value $\varepsilon_{i,j}$ used to "perturbate" the corresponding cost $c_{i,j}$ is obtained by randomly generating it (according to a uniform distribution) if $i,j \in N$, otherwise by setting it to zero.

The choice of the infeasible path $\mathcal{P}^{(\alpha)}$ to be broken at node α is performed as follows. Since the heuristic procedure we use to check the dominance criterion is more powerful when few depots exist from which imposed paths start, we try to keep the number of such depots as small as possible. This is obtained by choosing, at each node α, the infeasible path $\mathcal{P}^{(\alpha)}$ among those starting from the depot, D_l, which has the maximum number of arcs in the imposed paths starting from it. If more than one infeasible path starts from D_l, path $\mathcal{P}^{(\alpha)}$ is chosen as that with the minimum number of non-imposed arcs (so as to reduce the number of descendent nodes). Ties are broken by selecting the path visiting the vertex having the maximum value of $\delta^{(h)}$ among all vertices h belonging to the equivalent paths.

Since all vertices belonging to the same set W_k are equivalent, in the sense that they can be interchanged without altering the cost of the feasible solutions, when arc (v_1,v_2) [resp. (v_h,v_{h+1})] with $v_1 \in W_{k_1}$ and $v_{h+1} \in W_{k_2}$, is forbidden, all arcs (i, v_2) with $i \in W_{k_1}$ [resp. (v_h, j) with $j \in W_{k_2}$] are forbidden as well. In this way, obvious equivalences are avoided.

At each node of the branch decision tree a simple interchange heuristic for MD–VSP is applied to anticipate updating of the incumbent solution. Let i and j be two vertices in N belonging to two infeasible paths, $\mathcal{P}_1$ and $\mathcal{P}_2$, of the optimal solution to AP, such that $\mathcal{P}_1$ starts from the same depot, D_l, at which $\mathcal{P}_2$ finishes. Let k be the vertex following i in $\mathcal{P}_1$, and h that preceding j in $\mathcal{P}_2$. The number of infeasible paths is reduced by at least one by interchanging arcs (i,k) and (h,j) with (i,j) and (h,k), with an extra-cost $\Delta_{i,j} = (c_{i,j} + c_{h,k}) - (c_{i,k} + c_{h,j})$ [$\Delta_{i,j}$ is ∞ when (T_i,T_j) or (T_h,T_k) are not compatible pairs). The interchange heuristic iteratively reduces the number of infeasible paths by performing the arc interchange associated with the minimum extra-cost $\Delta_{i,j}$ (ties are broken by selecting, if possible, paths $\mathcal{P}_1$ and $\mathcal{P}_2$ so that the depot at which $\mathcal{P}_1$ finishes is the same as that from which $\mathcal{P}_2$ starts, thus reducing by two the number of infeasible paths). The overall time complexity of the interchange heuristic is $O(n^3)$ time. In our implementation, we consider only arcs (i,j) having zero reduced cost after the AP solution.

Computation of the optimal solution of the assignment problem associated with each node of the branch decision tree is performed through parametric techniques as described in Carpaneto and Toth [5].

VII. COMPUTATIONAL RESULTS

The branch-and-bound algorithm of the previous section has been implemented in FORTRAN 77 and run on an HP 9000/840 computer.

We have considered test problems randomly generated so as to simulate real-life public transport instances. Let $\rho_1,\rho_2, \ldots ,\rho_\nu$ be the *relief points* (i.e., the points where trips can start or finish) of the transport network. We have generated them as uniformly random points in a (60×60) square and computed the corresponding travel times $\theta_{a,b}$ as the Euclidean distances between relief points a and b. As for the trip generation, we have generated for each trip $T_j (j = 1, \ldots ,n)$ the starting and ending relief points, (ρ'_j) and (ρ''_j), as uniformly random integers in $(1,\nu)$. We have then $\tau_{i,j} = \theta_{\rho''_i,\rho'_j}$ for each pair of trips T_i and T_j. The starting and ending times, s_j and e_j, of trip T_j have been generated by considering two classes of trips: *short trips* (with probability 40%) and *long trips* (with probability 60%).

i) Short trips: s_j as uniformly random integer in (420,480) with probability 15%, in (480,1020) with probability 70%, and in (1020,1080) with probability 15%, e_j as uniformly random integer in $(s_j + \tau_{\rho'_j,\rho''_j} + 5,\ s_j + \tau_{\rho'_j,\rho''_j} + 40)$.

ii) Long trips: s_j as uniformly random integer in (300,1200) and e_j as uniformly random integer in $(s_j + 180, s_j + 300)$.

In addition, for each long trip T_j we impose $\rho''_j = \rho'_j$.

In this way, we simulate a real-life public transport instance in which times are given in minutes, and we have both short and long trips running from 5 a.m. to around midnight. Long trips correspond to extra-urban journeys, or to sequences of urban journeys. Short trips correspond to urban journeys and are generated so as to determine peak hours around 7–8 a.m. and 5–6 p.m.

As for the depots, we have considered two values of $m (m = 2,3)$ and two classes of problems, A and B. For class A, all the m depots have been randomly located at points inside the (60×60) square. For class B, in the $m = 2$ case we have located the two depots at the opposite corners of the square; in the $m = 3$ case, the third depot has been randomly located at a point inside the square. The number r_k of vehicles

TABLE I. Problems of class A ($m = 2$). 10 problems solved for each entry. HP 9000/840 sec.

	Algorith BB				Algorith BBL				Algorithm BBLD			
n	Aver. time	Max time	Aver. nodes	Aver. gap %	Aver. time	Max time	Aver. nodes	Aver. gap %	Aver. time	Max time	Aver. nodes	Aver. gap %
30	3.5	21.4	37.6	0.7	2.7	14.5	21.4	0.4	2.7	14.8	21.3	0.4
40	24.6	67.5	179.0	1.3	10.9	34.9	58.0	0.8	11.2	34.8	57.6	0.8
50	649.6*	4000	3948.2	0.8	467.2	3621.4	1721.5	0.5	463.2	3603.9	1529.2	0.5
60	1388.2*	4000	4854.3	1.2	994.4	3807.5	2391.1	0.8	734.5	3693.5	1789.1	0.8
70	1756.0*	4000	3351.7	1.1	1033.4*	4000	1490.2	0.6	1002.3*	4000	1362.0	0.6

(*) one instance exceeded the 4000 sec. time limit

TABLE II. Problems of class A ($m = 3$) and class B ($m = 2, 3$). Algorithm BBLD. 10 problems solved for each entry. HP 9000/840 sec.

	Class A ($m = 3$)				Class B ($m = 2$)				Class B ($m = 3$)			
n	Aver. time	Max time	Aver. nodes	Aver. gap %	Aver. time	Max time	Aver. nodes	Aver. gap %	Aver. time	Max time	Aver. nodes	Aver. gap %
30	12.2	58.2	175.1	1.6	4.9	27.1	43.9	1.0	10.2	58.2	65.9	0.9
40	63.6	264.4	209.2	0.8	19.6	63.6	102.6	0.8	147.1	685.1	679.5	1.0
50	563.5*	4000	2386.2	1.1	512.3	3621.3	1557.2	0.9	616.9	3861.8	1566.2	0.8
60	1938.3**	4000	4997.7	1.3	898.4*	4000	2330.3	0.5	1811.6**	4000	6321.2	1.6
70	—	—	—	—	1307.9*	4000	2073.1	0.6	—	—	—	—

(*) One instance exceeded the 4000 sec. time limit
(**) Two instances exceeded the 4000 sec. time limit.

stationed at each depot D_k has been generated as uniformly random integer in $(3 + n/(3m), 3 + n/(2m))$. This generation generally ensures feasibility of the instance.

Costs $\gamma_{i,j}$, $\bar{\gamma}_{k,j}$ and $\tilde{\gamma}_{j,k}$ have been obtained as follows:

i) $\gamma_{i,j} = \lfloor 10\, \tau_{i,j} + 2(s_j - e_i - \tau_{i,j}) \rfloor$, for all compatible pairs (T_i,T_j);

ii) $\bar{\gamma}_{k,j} = \lfloor 10$ (Euclidean distance between D_k and ρ'_j)$\rfloor + 5000$, for all depots D_k and trips T_j;

iii) $\tilde{\gamma}_{j,k} = \lfloor 10$ (Euclidean distance between ρ''_j and D_k)$\rfloor + 5000$, for all trips T_j and depots D_k.

The objective is to minimize the number of vehicles used in the solution and then minimize a mixture of travel and idle times of the vehicles.

Five values of n (30,40,50,60,70) have been considered for each value of m and each class. The corresponding value of v is a uniformly random integer in $(n/3,n/2)$. For each class and for each value of m and n, 10 instances have been generated and solved.

In order to evaluate the effectiveness of imbedding the improved lower bound of Section IV and the dominance criterion of Section V in the branch and bound algorithm of Section VI, BBLD, we derived two simplified versions, BBL and BB, of BBLD by removing the dominance procedure and both the dominance procedure and the lower bound improvement (steps 2–20 of ADD-MDVSP, see Section IV.B), respectively, at each node of the branch-decision tree.

Table I compares the performances of algorithms BB, BBL, and BBLD for problems of class A with $m = 2$. For each value of n, the table gives the average running time, the maximum running time (both in HP 9000/840 seconds), the average number of nodes of the branch-decision tree, and the average percentage gap between the optimal solution value and the lower bound at the root node. For each instance and for each algorithm, a time limit of 4,000 seconds was imposed (when this time was exceeded, execution of the algorithm was interrupted and the current state stored for statistics).

Table I shows that both the lower bound improvement and the dominance criterion are worth computing, since they significantly reduce the number of nodes, as well as global running time.

Table II gives the average performance of algorithm BBLD for problems of class A with $m = 3$, and of class B with $m = 2,3$. The table shows that problems with $m = 3$ are more difficult than those with $m = 2$, due to the lower bound quality and to the presence of a larger number of infeasible paths in the former case. As for the depots location, no significant differences between classes A and B result.

ACKNOWLEDGMENT

This work was supported by C.N.R., Italy.

References

[1] A. A. Bertossi, P. Carraresi, and G. Gallo, On some matching problems arising in vehicle scheduling models. *Networks* **17** (1987) 271–281.

[2] L. Bodin and B. Golden, Classification in vehicle routing and scheduling. *Networks* **11** (1981) 97–108.

[3] L. Bodin, B. Golden, A. Assad, and M. Ball, Routing and scheduling of vehicles and crews: The state of the art. *Computers & Operations Res.* **10** (1983) 63–211.

[4] L. Bodin, D. Rosenfield, and A. Kydes, UCOST. A micro approach to a transit planning problem. *J. Urban Anal.* **5** (1978) 47–69.

[5] G. Carpaneto and P. Toth, Some new branching and bounding criteria for the asymmetric travelling salesman problem. *Management Sci.* **26** (1980) 736–743.

[6] P. Carraresi and G. Gallo, Network models for vehicle and crew scheduling. *Eur. J. Operational Res.* **16** (1984) 139–151.

[7] P. Carraresi and G. Gallo, Optimization models in mass transit resources management. *Ricerca Operativa* **38** (1986) 121–150.

[8] M. Fischetti and P. Toth, An additive bounding procedure for combinatorial optimization problems. *Operations Research* **37** (1989).

[9] M. Fischetti and P. Toth, A new dominance procedure for combinatorial optimization problems. *Operations Research Letters* **7** (1988) 181–187.

[10] B. Smith and A. Wren, VAMPIRES and TASC: Two successfully applied bus scheduling programs. In *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling*, A. Wren (Ed.). North-Holland, Amsterdam (1981) 97–124.

[11] A. Wren, General review of the use of computers in scheduling buses and their crews. In *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling*, A. Wren (Ed.). North-Holland, Amsterdam (1981) 3–16.

Received May 1987
Accepted August, 1988