Invited Review

Network models for vehicle and crew scheduling

P. CARRARESI and G. GALLO
Department of Informatics, University of Pisa, Italy

Received October 1983

Network models and methods have proven to be rather successful in many application fields. In this survey their use in the solution of Vehicle Scheduling and Crew Scheduling problems in Mass Transit settings is reviewed. An attempt is made to encompass in a unified framework some of the most relevant models on the subject to be found in the literature.

1. Introduction

Since the pioneering work of Ford and Fulkerson [18], the Network Flow Theory has enjoyed rapidly increasing acceptance within the scientific community, to such an extent as to become one of the most important sectors of the Mathematical Programming field. Correspondingly, the use of Network models and algorithms has proven to be particularly successful in many different application areas.

In this survey we address ourselves to the use of network methods in solving a class of problems of great practical importance: *Vehicle and Crew Scheduling in Mass Transit settings*.

The need for mathematical methods in solving these problems has increased over the last few years in connection with the widespread use of computers in transportation companies. Many attempts. some of them quite successful, have been made to develop new methods and to design computer codes for Vehicle and Crew Scheduling problems [32,34]. Some of them, mainly the earlier ones, are based on heuristic techniques, often derived from the manual procedures traditionally used in the companies. Others, though using some heuristic procedures, also make use of more sophisticated network-based optimization algorithms. We shall review the approaches of this latter type within a unified framework.

The survey is organized into three sections devoted to the following problems: *Vehicle Scheduling, Crew Scheduling* and *Rostering*, in that order. In the selection we have made, only papers focusing on network models and methods have been included, thus leaving out quite a few valuable contributions in which network-derived ideas are not essential to the proposed approaches.

2. Vehicle scheduling

In this paper the term 'Vehicle Scheduling' refers to that large class of optimization problems in which vehicles must be assigned to time-tabled trips in such a way that:

(i) each trip is carried out by one vehicle;
(ii) a given set of constraints is satisfied;
(iii) a properly defined cost function is minimized.

We assume in the following that a time-table is given, that is a set $V = \{v_1, v_2, \dots, v_n\}$ of trips, where each *trip* v_i is defined by a quadruple (τ_i, l_i, o_i, d_i), with

τ_i, the departure time;
l_i, the length of time, or duration;
o_i, the origin, or departure terminal;
d_i, the destination, or arrival terminal;

In addition to regular trips, deadheading trips are allowed between different terminals; by δ_{ij} we denote the duration of a deadheading trip from d_i to o_j.

The ordered pair (v_i, v_j) is said to be a *compatible pair of trips* if

$$\tau_i + l_i + \delta_{ij} + \epsilon \leqslant \tau_j, \tag{2.1}$$

where $\epsilon \geqslant 0$ is a prefixed tolerance parameter, introduced to take possible delays into account. Inequality (2.1) states that it is feasible to have trips v_i and v_j operated in sequence by the same vehicle.

Furthermore a subset $C = \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$ of the set V is said to be a *vehicle duty* if $(v_{i_j}, v_{i_{j+1}})$ is a compatible pair of trips, $j = 1, \ldots, k-1$: the trips of the set C can be run by the same vehicle. Then a feasible *Vehicle Schedule* can be defined as a family $\{C_1, C_2, \ldots, C_r\}$ of vehicle duties, such that each trip $v \in V$ belongs to exactly one C_j, $j = 1, \ldots, r$.

With each Vehicle Scheduling problem we can associate a simple and extremely useful graph. Let $G = \{S, T, A\}$ be a bipartite graph where $S = \{s_1, s_2, \ldots, s_n\}$ and $T = \{t_1, t_2, \ldots, t_n\}$ are the sets of nodes and $A = \{(s_i, t_j)$: (v_i, v_j) is a compatible pair$\}$ is the set of arcs.

A matching on G is a subset of arcs $\bar{A} \subseteq A$ such that each node is incident to at most only one arc of $\bar{A}$.

Proposition 2.1. *There is a one-to-one correspondence between Vehicle Schedules and matchings on G.*

Proposition 2.1 follows from the fact that any vehicle duty, say $C_i = \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$, can be represented in G as the set of arcs $\bar{A}_i = \{(s_{i_1}, t_{i_2}), (s_{i_2}, t_{i_3}), \ldots, (s_{i_{k-1}}, t_{i_k})\}$, where arc $(s_{i_j}, t_{i_{j+1}})$ corresponds to the deadheading trip from d_{i_j} to $o_{i_{j+1}}$, $j = 1, \ldots, k-1$. Since for each $v_{i_j} \in C_i$, at most one arc in $\bar{A}_i$ is incident to nodes s_{i_j} and t_{i_j}, and since each trip belongs to only one of the vehicle duties $C_1, \ldots, C_r$, the set $\bar{A} = \bar{A}_1 \cup \cdots \cup \bar{A}_r$ is a matching. Similarly a matching $\bar{A}$ of G can be interpreted as a Vehicle Schedule with the following proviso: if no arc of $\bar{A}$ is incident either to s_j or to t_j, then a vehicle runs the single trip v_j, i.e. a vehicle duty exists containing only the trip v_j.

For instance consider a set of 5 trips described in Table 1, where T_a, T_b, T_c, T_d denote four terminals with duration (in minutes) of deadheading trips given in Table 2. Then the corresponding graph G is given in Fig. 1. As an example the matching $\{(s_1, t_3), (s_2, t_4), (s_4, t_5)\}$ on the graph G in Fig. 1, corresponds to the two vehicle duties: $\{v_1, v_3\}$ and $\{v_2, v_4, v_5\}$.

Table 1

v_i	τ_i	l_i	o_i	d_i
v_1	7:10	20	T_a	T_b
v_2	7:20	20	T_c	T_d
v_3	7:40	25	T_b	T_a
v_4	8:00	30	T_d	T_c
v_5	8:35	30	T_c	T_d

Table 2

	T_a	T_b	T_c	T_d
T_a	0	15	20	20
T_b	15	0	25	10
T_c	20	25	0	15
T_d	20	10	15	0

2.1. *Basic networks models*

From the bipartite graph (S, T, A) a simple network model can be derived, which represents the core of all the models of the scheduling problems we shall consider in the following sections.

Define the network $N = (S, T, A \cup A^*)$, where s_i, $i = 1, \ldots, n$, are source nodes with unit input flow, t_j, $j = 1, \ldots, n$, are sink nodes with unit demand and $A^* = \{(s_i, t_j)$: $(s_i, t_j) \notin A$, and either $\tau_i \geqslant \tau_j + l_j + \delta_{ji}$, or $i = j\}$. A feasible flow is a vector x, with one component x_{ij} for each arc $(s_i, t_j) \in A \cup A^*$, satisfying the following constraints:

$$\begin{aligned} &\sum_{j \in J(i)} x_{ij} = 1, && i = 1, \ldots, n, \\ &\sum_{i \in I(j)} x_{ij} = 1, && j = 1, \ldots, n, \\ &x_{ij} \geqslant 0, && j \in J(i), i = 1, \ldots, n, \end{aligned} \tag{2.2}$$

where $J(i) = \{j$: $(s_i, t_j) \in A \cup A^*\}$ and $I(j) = \{i$: $(s_i, t_j) \in A \cup A^*\}$.

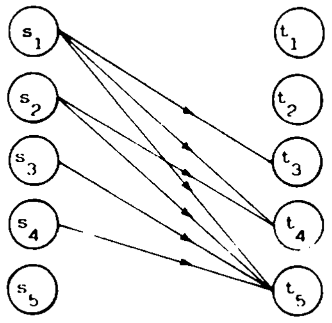

Fig. 1.

Let X be the set of integer feasible flows, that is the set of vertices of the polyheadron defined by constraints (2.2). Obviously X is not empty as it contains at least the solution $x_{ii} = 1$, $i = 1,\dots,n$, and $x_{ij} = 0$, $i \neq j$.

A vector $x \in X$ defines a assignment between S and T: the only node t_j for which $x_{ij} = 1$ corresponds in T to each node $s_i \in S$, and viceversa. A Vehicle Schedule corresponds to each vector $x \in X$ as follows: for each $x_{ij} = 1$, v_i and v_j belong in the order to the same vehicle duty, unless $(s_i, t_j) \in A^*$ in which case v_i and v_j are the last and the first trip respectively of a vehicle duty; in that case they may or may not belong to the same vehicle duty. If $x_{ii} = 1$ for some i then trip v_i is the only trip of a vehicle duty. conversely it is possible to show that an element $x \in X$ corresponds to each Vehicle Schedule. It is worth noting that the correspondence is not of the one-to-one type. In fact, the same Vehicle Schedule might correspond to distinct vectors of X which differ only by the flows on arcs of A^*; on the contrary if the vectors differ also by the flows on arcs of A, then they correspond to distinct Vehicle Schedules. From these considerations the following proposition derives:

Proposition 2.2. *Each integral feasible flow (assignment) on N corresponds to a feasible Vehicle Schedule where the number of vehicles is equal to the number of units of flow on arcs of the set A^*.*

Then, at least in most cases, a vehicle scheduling problem can be formulated as the problem of finding an integral feasible flow on N (possibly satisfying additional constraints depending on the particular problem) minimizing a given objective function. In general the objective function is to minimize the cost of the schedule. Two types of costs are usually considered: capital cost and operational cost. The former is strictly dependent on the fleet size and characteristics, i.e. number and type of vehicles. The latter depends mainly on deadheading trips and idle times between two consecutive trips. It is also possible to consider weighted combinations of these two costs.

A variant of the model is obtained by considering the slightly different network $N' = (S', T', A')$, where $S' = S \cup \{s_0\}$, $T' = T \cup \{t_0\}$ and $A' = A \cup \{(s_i, t_0)\colon\ i = 1,\dots,n\} \cup \{(s_0, t_j)\colon\ j = 1,\dots,n\} \cup \{(t_0, s_0)\}$. Nodes s_0 and t_0 are intermediate nodes, while S and T are the sets of the sources and the sinks respectively, as in N.

The main difference with respect to the previous model is that an arc $(s_i, t_j) \in A^*$ is replaced by the path $((s_i, t_0), (t_0, s_0), (s_0, t_j))$. The advantage, at least for problems with large sets A, is in the smaller number of arcs: in fact the set A^*, whose cardinality can be quite large is replaced by $2n + 1$ arcs. This fact could result in significant saving either in computer storage or in CPU time, when dealing with large-scale problems. It is worth noting that this second model is more adherent to the physics of the problem: in fact nodes s_0 and t_0 can be considered as corresponding to the depot, and flows from s_0 and to t_0 correspond to vehicles leaving from the depot to start their duty and returning to it after completion of the duty respectively.

2.2. The single depot problem

Here we consider the case in which all vehicles are housed at the same depot. This is the most frequent case with small size transit companies; it is also the case with large ones when, as quite often happens, the serviced area is partitioned into zones, each of them assigned to one single depot.

Furthermore we assume that no constraint is present other than the compatibility constraint defined by (2.1), and that all vehicles are of the same type. These are fairly reasonable assumptions at least for urban and suburban transportation. It is true that in some cases more than one type of vehicle is used, but usually it is decided a priori which trip is operated by which vehicle type: then the problem splits into subproblems, one for each vehicle type.

With these assumptions the problem becomes the rather simple assignment problem

$$\operatorname{Min}\{cx\colon x \in X\} \tag{2.3}$$

where c is the vector of the arc costs and depends on the particular objective we choose.

If we want to minimize the capital cost we have to set

$$c_{ij} = \begin{cases} 1, & \text{if } (s_i, t_j) \in A^*, \\ 0, & \text{otherwise.} \end{cases}$$

In fact minimizing the capital cost is equivalent to minimizing the fleet size.

If we want to minimize the operational costs, then the following arc costs will do:

$$c_{ij} = \begin{cases} \text{the cost of deadheading traveling and idle time between the end of trip } v_i \text{ and the beginning of trip } v_j \text{ if } (s_i, t_j) \in A; \\ \text{the cost of deadheading trips from the depot to the departure terminal of } v_j \text{ plus the cost of the deadheading trip from the arrival terminal of } v_i \text{ to the depot if } (s_i, t_j) \in A^*. \end{cases}$$

Note that in the case of long idle times it might be cheaper to send the vehicle to the depot after trip v_i; the vehicle will resume its schedule later with trip v_j; in evaluating cost c_{ij} this fact should be considered.

Clearly one can minimize a combination of the two costs. Furthermore if one seeks the minimum operational cost with the constraint that the number of vehicles be minimum, it is enough to add to the costs of arcs in A a suitably-sized constant.

Independently of the choice for c, problem (2.3) is a rather simple problem which can be solved efficiently by making use of either assignment or min-cost flow codes [11,22,31]. Formulation (2.3) of the vehicle scheduling problem as been widely reported in the literature [20,21,30], although not always fully exploited in applications.

2.3. The multiple depot problem

The problem considered here is the same as the one in the preceding section except that l ($\geqslant 2$) depots exist, each with a given capacity (maximum number of vehicles housed).

The flow model in this case is of the multicommodity type; hence the resulting optimization problem is much more difficult than the single depot problem and it is unlikely that polynomial algorithms can be found to solve it exactly [23].

The multiple depot problem can be formulated as follows:

$$\text{Min} \sum_{k=1}^{l} \sum_{(i,j) \in A \cup A^*} c_{ij} x_{ij}^k,$$

s.t.

$$\sum_{j \in J(i)} x_{ij}^k = y_{ik}, \quad i = 1,\ldots,n, \; k = 1,\ldots,l,$$

$$\sum_{i \in I(j)} x_{ij}^k = y_{jk}, \quad j = 1,\ldots,n, \; k = 1,\ldots,l,$$

$$\sum_{(i,j) \in A^*} x_{ij}^k \leqslant a_k, \quad k = 1,\ldots,l, \tag{2.4}$$

$$\sum_{k=1}^{l} y_{ik} = 1, \quad i = 1,\ldots,n,$$

$$x_{ij}^k \geqslant 0, \quad (i,j) A \cup A^*, \; k = 1,\ldots,l,$$

$$y_{ik} \in \{0, 1\}, \quad i = 1,\ldots,n, \; k = 1,\ldots,l$$

where a_k denotes the capacity of depot k. The variables in (2.4) have the following meaning ($k = 1, 2,\ldots,l$):

- for $(i,j) \in A$

$$x_{ij}^k = \begin{cases} 1, & \text{if a vehicle housed at depot } k \text{ runs trips } v_i \text{ and } v_j \text{ in sequence,} \\ 0, & \text{otherwise;} \end{cases}$$

- for $(i,j) \in A^*$

$$x_{ij}^k = \begin{cases} 1, & \text{if trips } v_i \text{ and } v_j \text{ are the first and the last trip respectively of vehicle duties assigned to depot } k, \\ 0, & \text{otherwise;} \end{cases}$$

- for $i = 1,\ldots,n$

$$y_{ik} = \begin{cases} 1, & \text{if trip } v_i \text{ is assigned to a vehicle housed at depot } k, \\ 0, & \text{otherwise.} \end{cases}$$

Note that there is no need to impose the integrality constraints on variables x_{ij}^k; in fact once the y_{ik} are fixed (at binary values), the problem decomposes into l smaller assignment problems with integer solutions.

The multi-depot problem is usually solved heuristically by decomposing it into simpler subproblems, which can be solved by means of polynomially bounded algorithms.

Two alternative heuristic approaches are described below [9,10].

(i) *Cluster first – Schedule second.* The set of trips is partitioned into l subsets, one for each depot. Then for each subset a single-depot problem is solved. If the capacity constraint for some depot is not satisfied, the partition is modified and new single depot problems are solved. The process continues until a satisfactory solution is obtained. This approach corresponds to the way in which most companies operate in practice, since quite often the trips are assigned to the depots a priori.

(ii) *Schedule first – Cluster second.* First a single-depot problem on the whole set of trips V is solved, yielding a set of vehicle duties. Then each vehicle duty is assigned to one of the depots in such a way that the total cost of deadheading trips be minimum and the capacity constraints be satisfied. This second optimization problem is a mimum cost network flow on the network of Fig. 2, where nodes d_1 to d_l correspond to depots, and nodes 1 to r correspond to vehicle duties, arc $(0, d_k)$ has zero cost and capacity a_k, $k = 1,\dots,l$, arc (d_k, h), for all k and h, has unlimited capacity and cost equal to the cost of the deadheading trips from the kth depot to the departure terminal of the first trip in the hth vehicle duty and from the arrival terminal of the last trip in the hth vehicle duty to the kth depot.

The origin o has an input of r units of flow and each of the nodes corresponding to vehicle duties has requirement 1. This second approach seems more effective than the first one, where the partition of the set V is somewhat arbitrary and may result in poor schedules.

An integrated approach based on lagrangean relaxation and subgradient optimization techniques is currently under study [16]. Although still at an experimental stage, the first results seem to be quite promising.

2.4. Periodic schedules

Vehicle Scheduling problems, at least in general, are periodical in their nature, in the sense that after a given time period, say 24 hours, all the trips must be run again. So far we have disregarded this aspect because in most cases it is enough to determine an optimal schedule for a single day, then the same schedule is repeated identically for all the days without losing optimality. Usually more than one schedule is needed: one for weekdays, one for Saturday and one for Sunday. This is usually true when dealing with transit in urban areas where, except for a very few night trips, today's schedule does not affect tomorrow's one.

There are, however, cases in which we cannot disregard the periodicity if we want to find an optimal, or even a feasible solution. This happens with long distance transportation. Consider for instance the following simple example [29]: a company operates daily the three flights described in Table 3, with inter city travel times given in Table 4; the objective is to find a feasible schedule with a minimum number of vehicles. Clearly at least three airplanes are needed since no two flights exist which can be run sequentially the same day. Then we may decide to assign one airplane to each flight on the first day; but if the schedule is repeated the following day with each airplane always running the same flight we need a fourth airplane because a single airplane cannot operate flight No. 2 on two consecutive days. The resulting schedule is represented by the graph of Fig. 3 where node v_i^h represents the ith flight of day h, and an arc exists from node v_i^h to node v_j^k if the same airplane runs in the sequence the ith flight on day h and the jth flight on day k.

A solution with only three airplanes does nevertheless exist in this case, and is depicted in Fig. 4.

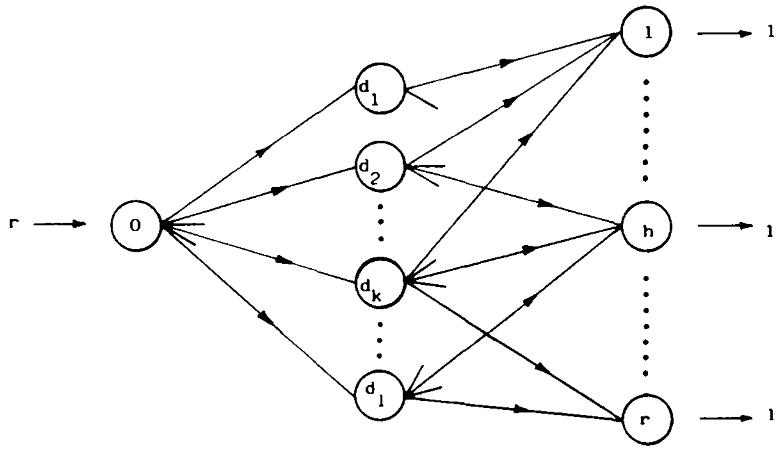

Fig. 2.

Table 3
Daily flights

Flight No.	Depart		Arrive	
1	Honolulu	1:00 p.m.	Washington DC	11:00 p.m.
2	New York	3:00 p.m.	Tokyo	4:00 a.m.
3	London	1:00 p.m.	Paris	2:00 p.m.

Table 4
Flight times in hours

	Honolulu	London	New York
Paris	15	1	7
Tokyo	8	12	13
Washington	10	7	1

In this second solution two airplanes run the first and the second flights alternatively.

We may now state formally the *Periodic Vehicle Scheduling* problem.

A set of trips $V = \{v_1, v_2, \dots, v_n\}$ is given, where each trip, v_i, must be run periodically every p units of time starting at time $\tau_i^0 = \tau_i$. We call p the period, and denote as v_i^h the re-run of trip v_i, starting at time $\tau_i^h = \tau_i + hp$.

Trip v_i is said to be q-compatible with trip v_j (possibly $j = i$) if

$$\tau_j + (q-1)p < \tau_i + l_i + \delta_{ij} + \epsilon \leqslant \tau_j + qp$$

where l_i, δ_{ij} and ϵ have their usual meaning. If trip v_i is q-compatible with trip v_j then the same vehicle can run sequentially v_i^h and v_j^k only if $k - h \geqslant q$.

With a periodic vehicle scheduling problem we can associate a graph $G = \{N, A\}$, where the set of nodes $N = \{1, \dots, n\}$ corresponds to the trip set V, and the set of arcs A corresponds to the pairs of compatible trips, i.e. $A = \{(i,j)$: v_i is q-compatible with v_j for some $q\}$. A weight $w(i,j) = q$ is assigned to arc (i,j) if v_i is q-compatible with v_j. The graph corresponding to the example in Tables 3 and 4 is given in Fig. 5 where the arcs carry their weights.

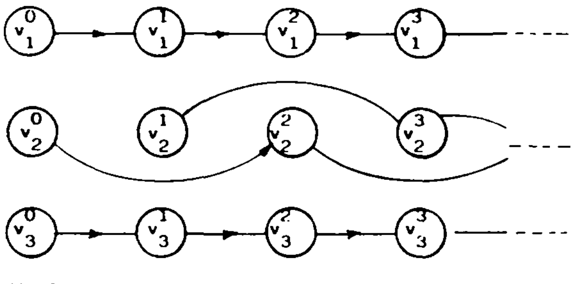
Fig. 3.

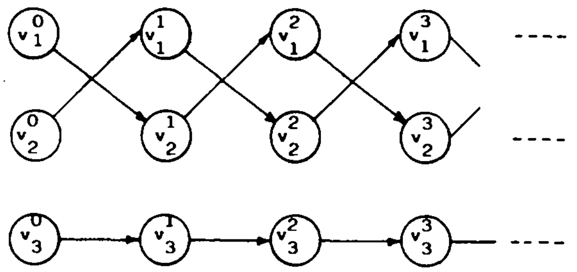
Fig. 4.

A *cycle cover* of a graph is a set of disjoint cycles which contain all the nodes. The weight of a cycle cover is the sum of the weights of the arcs in the cycles. The following result [29] provides us with a simple way to solve the periodic vehicle scheduling problem.

Proposition 2.3. *The minimum number of vehicles needed to run the trips of the set V is given by the minimum weight of the cycle covers in G.*

A minimum cycle cover of the graph in Fig. 5 is given in Fig. 6. It corresponds to the schedule illustrated in Fig. 4.

Needless to say, a minimum weight cycle cover can be determined solving an assignment problem on the graph $\{S, T, A\}$ where $S = \{s_1, \dots, s_n\}$, $T = \{t_1, \dots, t_n\}$, $A' = \{(s_i, t_j)$: $(i,j) \in A\}$ and the cost of arc (s_i, t_j) is equal to $w(i,j)$.

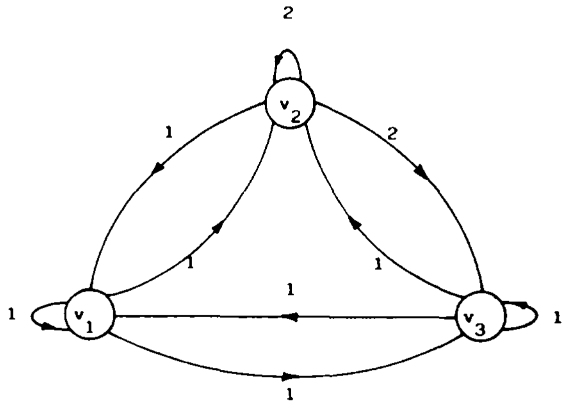
Fig. 5.

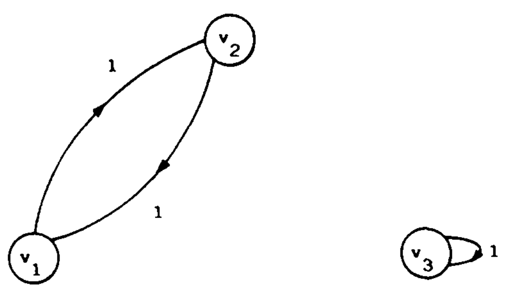

Fig. 6.

3. Crew scheduling

In this section we deal with the *Crew Scheduling Problem in a Mass Transit setting*, i.e. the problem of finding a set of duties to be assigned to drivers in order that the daily service requirement be met at minimum cost.

As in Section 2, our primary concern will be with the development of network models. Although some first attempts are being made in the direction of solving the vehicle and crew scheduling problems jointly [5,6], the two problems are usually solved sequentially. It can be added that the crew scheduling problem is in itself difficult enough, and that much work is still needed to find efficient algorithmic approaches capable to handle, at reasonable computational cost, large scale problems.

In the following we shall assume that the vehicle scheduling problem has already been solved, and that the set of vehicle duties defining the vehicle schedule is known. To avoid misunderstanding, from now on we shall use the term '*block*' as is often used in the literature, rather than 'vehicle duty'.

For each block a set of *relief times*, i.e. times at which a driver's substitution may occur, is given. A *piece of work* is a continuous driving period: we shall denote by $s(p)$ and $e(p)$ the starting time and the ending time respectively of piece of work p. A piece of work is feasible for a block if both $s(p)$ and $e(p)$ are relief times of the block. A *duty* consists of a set of pairs (p, k), where p is a piece of work and k the block on which the piece of work shall be run. Only duties which comply with a certain number of rules depending on the particular problem are accepted. A regular duty is one which can be assigned to a full-time driver as his daily work assignment; a regular duty made up of a single piece of work is called a *straight run*. Small duties containing only one short piece of work are also accepted in order to ensure feasibility; such duties, called *trippers*, are assigned either to part-time drivers or, as overtime, to full-time drivers. Usually the companies want to avoid the use of trippers, or at least to minimize their number, since their cost per unit of time is higher than the cost of regular duties.

Quite often, in solving the problem, additional constraints peculiar to particular situations have to be satisfied. More often than not these constraints derive from trade-union regulations.

Given the difficulty of the problem, it is not uncommon for some rather harsh simplifications to be made in solving it. In the following we will show that most models used in practice can be derived from quite a general and neat model which embeds network submodels.

3.1. The basic model

Let $T_k = \{t_1^k, t_2^k, \ldots, t_{u_k}^k\}$ and P_k, with $m_k = |P_k|$, denote the set of relief times and the set of pieces of work feasible for block k, respectively; t_1^k and $t_{u_k}^k$ are the starting and ending times of the block.

Introduce now the set $D = \{d_1, d_2, \ldots, d_{|D|}\}$ of all feasible duties. For the sake of simplicity we assume that only duties with either one or two pieces of work are allowed: a duty d made up of piece p on block k and piece q on block h will be denoted by $d = \{(p, k), (q, h)\}$ where $p = q$ and $k = h$ if d is a straight run.

A subset $\tilde{P}_k \subseteq P_k$ is a partition of block k if for each relief time t_i^k, $i = 1, 2, \ldots, u_k - 1$, there is one and only one $p \in \tilde{P}_k$ such that $s(p) \leqslant t_i^k < e(p)$: a partition of a block is a set of pieces of work which cover the service on that block.

The crew scheduling problem comes down to finding one partition for each block and then matching together pieces of work from the partitions in order to get feasible duties. Out of the many feasible solutions, we seek one which minimizes the total cost. We show next how this problem can be phrased in a rather neat and general graph model, then we will show how such a model can be simplified in order to get at reasonable computational cost good suboptimal solutions.

Associated with each block k, define the network $G_k = (N_k, A_k)$, where $N_k = T_k$ and $A_k = \{(s(p), e(p)):\ p \in P_k\}$; i.e. each node corresponds to one of the relief times, while each arc corresponds to one of the pieces of work feasible for the block; t_1^k and $t_{u_k}^k$ are the origin and the destination of the network respectively.

For instance, consider a block with relief times at 8:30, 9:30, 10:20, 11:20 and 12:30, and assume that feasible pieces of work have lengths between 1 and 2 hours; then the associated network is given in Fig. 7.

A partition of block k corresponds to a path from node t_1^k to node $t_{u_k}^k$ in G_k and viceversa. In the example of Fig. 7 the path (8:30, 9:30, 11:20, 12:30) corresponds to the partition $\bar{P}_k = \{(8{:}30, 9{:}30), (9{:}30, 11{:}20), (11{:}20, 12{:}30)\}$.

Then the problem of finding a partition of a block is to find an origin-destination path on the corresponding network. Such a problem can be formulated as follows:

$$- \sum_{\substack{p\in P_k\\ e(p)=l}} y_p^k + \sum_{\substack{p\in P_k\\ s(p)=l}} y_p^k = \begin{cases} 1, & l = t_1^k, \\ 0, & l = t_i^k, i = 2,\dots,u_k - 1, \\ -1, & l = t_{u_k}^k, \end{cases} \tag{3.1}$$

$$y_p^k \in \{0, 1\}, \quad p \in P_k,$$

where

$$y_p^k = \begin{cases} 1, & \text{if the arc corresponds to a piece of work used in the path,} \\ 0, & \text{otherwise.} \end{cases}$$

Constraints (3.1) can be rewritten in a more compact form as follows:

$$E^k y^k = b^k, \qquad y^k \in \{0, 1\}^{m_k}, \tag{3.2}$$

where E^k is the incidence matrix of G_k, b^k is an u_k-vector with $b_1^k = 1$, $b_{u_k}^k = -1$ and $b_i^k = 0$ otherwise and $m_k = |A_k|$.

Starting from D, a graph G can be defined, where G contains one node for each feasible piece

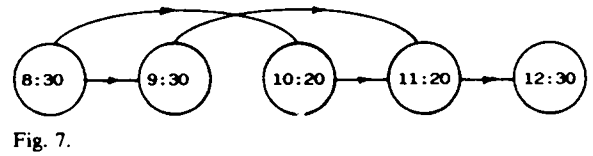

Fig. 7.

(p, k) such that $p \in P_k$, $k = 1,\dots,r$, and one arc for each feasible duty $d = \{(p, k), (q, h)\}$, connecting the two nodes corresponding to (p, k) and (q, h) respectively. A straight run corresponds in G to a self-loop.

A vector y^k satisfying (3.2) defines a partition of block k; then a m-vector $y = (y^1, y^2,\dots,y^r)$, with $m = \sum_k m_k$, corresponds to a set of pieces of work covering the service on all the blocks.

Consider the subgraph of G, $G(y)$, induced by the nodes corresponding to pieces (p, k) such that $y = 1$. The problem of deriving a schedule, starting from the pieces of work defined by the vector y, corresponds in $G(y)$ to finding a perfect matching, that is a subset of arcs such that each node is incident at exactly one arc in the subset.

Let x be a $|D|$-vector of binary variables corresponding to the set of all feasible duties, then the crew scheduling problem can be written as follows:

Min cx,

s.t.

$$\begin{array}{ll} \text{(i)} & E^k y^k = b^k, \\ \text{(ii)} & \sum_{j\in I_{pk}} x_j = y_p^k, \quad p \in P_k, k = 1,\dots,r, \\ \text{(iii)} & y^k \in \{0, 1\}^{m_k}, \\ \text{(iv)} & x \in \{0, 1\}^{|D|}, \\ \text{(v)} & x \in X \end{array} \tag{3.3}$$

where c is the cost vector, and I_{pk} is the subset of all the duty indices corresponding in G to arcs incident to node (p, k). Clearly only the arcs (duties) with the corresponding $x_j = 1$ will belong to the matching (schedule). Constraint (v) is added to take into account peculiarities of the various individual problems. Usually they can be described by means of a very small number of linear equalities or inequalities.

Problem (3.3) is a rather difficult one, because of the large amount of integer variables and constraints, also for small size transportation companies. Usually one has to simplify the problem in order to be able to solve it; of course the solutions so obtained are suboptimal.

The commonest techniques to simplify the problem are:

(i) introduction of proper assumptions to reduce the size;

(ii) aggregation of constraints: two or more constraints are added together to get a new aggregate constraint which replaces them;

(iii) linear transformation of variables: this is done to reduce the number of variables;

(iv) relaxation: one or more constraints are dropped;

(v) lagrangean relaxation: one or more constraints are moved into the objective function after multiplication by a proper penalty coefficient.

In the following sections we will illustrate different models which can be derived from (3.3) by means of proper simplifications. We will show how network optimization techniques are essential to their efficient solution.

3.2. *The set partitioning model*

Set partitioning models [3] have been widely used in the solution of the crew scheduling problem [26,27,28].

A quite general set partitioning formulation of the crew scheduling problem is

$$\begin{aligned}&\text{Min } cx,\\ &B^k x = a^k, \quad k = 1,\dots,r, \qquad (3.4)\\ &x \in \{0,1\}^{|D|},\end{aligned}$$

where B^k is a $u_k \times |D|$ matrix with components

$$b_{ij}^k = \begin{cases} 1, & \text{if } d_j \text{ contains the piece } (p,k) \\ & \text{with } s(p) \leqslant t_i^k < e(p), \\ 0, & \text{otherwise} \end{cases}$$

and a^k is a u_k-vector with components equal to 1.

It is easy to check that if we disregard constraints (3.3(v)), problem (3.4) can be derived from (3.3) via simple transformations: first constraints $E^k y^k = b^k$, $k = 1,\dots,r$, are premultiplied (both sides) by the non singular $(u_k \times u_k)$ matrix L^k with

$$l_{ij}^k = \begin{cases} 1, & i \geqslant j, \\ 0, & \text{otherwise;} \end{cases}$$

then the y variables are eliminated by means of (3.4(ii)).

The main difficulty when using a set partitioning approach is in the enormous amount of duties that need to be considered to ensure an optimal solution. In practice one takes into consideration only a small subset of D and/or tries to decompose the problem into a sequence of smaller subproblems. Successful applications are reported in the literature, mainly in the context of airline crews [1,2,4,26,27].

Needless to say the problem becomes much more difficult if constraints (3.3(v)) are considered; with these constraints, (3.4) becomes a general integer linear programming problem.

3.3. *The aggregate model*

A rather coarse relaxation of (3.3) is represented by the aggregate model, used to get either a starting approximate solution or a lower bound to the value of the objective function [25].

Assume that the day be partitioned into $u-1$ intervals $[t_{i-1}, t_i]$, $i = 2,\dots,u$, that only times $t_1, t_2,\dots,t_u$ are allowed as relief times and that if $t_i \in T_k$ and $t_j \in T_k$, with $i<j$, then $t_h \in T_k$ for each $h \in [i,j]$. These assumptions can be made reasonably realistic by choosing a large enough value for u and by making the widths of the intervals vary along the day, e.g. using the shortest intervals in the peak hours when the substitutions of drivers are more frequent.

The main effect of this assumption is to reduce the number of possible duties. Moreover if a piece of work is feasible for one block, it is also feasible for all the blocks. If we assume that the interval between two pieces of work in the same duty is always large enough to allow for transfers, we may consider the duties as defined by their pieces of work independent of the blocks; Then by introducing the set $J(p,q) = \{j\colon d_j$ contains pieces of work p and $q\}$ ($J(p,p)$ for a straight run made by the single piece of work p), we may define the new variables

$$z(p,q) = \sum_{J(p,q)} x_j. \qquad (3.5)$$

Since the time intervals corresponding to the rows in (3.4) are now the same for all the blocks, we may add up all the rows corresponding to the same time interval and then apply variable transformation (3.5), setting the following aggregate model:

$$\begin{aligned}&\text{Min } cz,\\ &\tilde{B}z = \tilde{a},\\ &z \in Z, \qquad (3.6)\\ &z \geqslant 0, \text{ integer},\end{aligned}$$

where z is the vector with components (3.5) and Z corresponds to the set X in (3.3).

Problem (3.6) is a relaxation of (3.4) and hence of the basic model (3.3). Then the solution it provides is, at least in general, not feasible for (3.4). Nevertheless it provides fairly accurate estimates of the cost of the optimal schedule, and a solution which can be useful used as a starting point for heuristic procedures. In practice (3.6) is solved by means of linear programming codes, disregarding the integrality constraints, and then an integer solution is obtained by means of rounding procedures [13].

The main difficulty with problem (3.6) is that, although it yields a solution with the required number of drivers for each time interval, it disregards the fact that the service occurs on different vehicles and that each piece of work must be done on the same vehicle from beginning to end. Additional constraints can be added to (3.6) to improve the feasible of the solution, but, at least in general, some further work has to be done to derive a feasible schedule from the optimal solution of (3.6) [24,25].

3.4. The matching model

Assume that an optimal solution $\bar{y}$ to (3.3) be known, then, if we disregard constraint (3.3(v)), the crew scheduling problem comes down to the matching problem

$$\begin{aligned} &\operatorname{Min} xc, \\ &\sum_{j \in I_{pk}} x_j = \bar{y}_p^k, \quad p \in P, k = 1, \dots, r, \qquad (3.7) \\ &x \in \{0, 1\}^{|D|}. \end{aligned}$$

Rather efficient algorithms exist to solve such a problem [19]. Very accurate, although in general not optimal, solutions can also be obtained by relaxing the matching problem (3.7) into a simpler assignment problem [17].

Matching models can be found in the literature as components of decomposition approaches to the crew scheduling problem, where first a reasonably good vector y is determined and then the matching problem (3.7) is solved [7,8,15].

The aggregate model combined with the matching one provides a convenient and not exceedingly costly way to get good solutions. Let $\bar{z}$ be the optimal solution of (3.6), and assume that a binary vector $\bar{y} = (\bar{y}^1, \bar{y}^2, \dots, \bar{y}^r)$ exists such that

$$\begin{aligned} &\sum_k \bar{y}_p^k = \sum_{q \in P} \bar{z}(p, q), \quad p \in P, \qquad (3.8) \\ &E^k \bar{y}^k = b^k, \quad k = 1, \dots, r, \end{aligned}$$

where P is the set of all feasible pieces of work, and the notation used in the same as in Section 3.2. Then it is easy to check that $\bar{z}$ is feasible and corresponds to an optimal schedule; the vector $\bar{y}^k$ defines the partition of block k in the schedule, $k = 1, \dots, r$.

From these considerations it follows that a reasonable way to find a feasible schedule starting from $\bar{z}$ is the following [13]:

(i) Solve the problem

$$\operatorname{Min} \sum_p \left(\sum_k y_p^k - \sum_q \bar{z}(p, q) \right)^2,$$

s.t.

$$\begin{aligned} &E^k y^k = b^k, \quad k = 1, \dots, r, \qquad (3.9) \\ &y_p^k \in \{0, 1\}, \quad p \in P, k = 1, \dots, r, \end{aligned}$$

yielding the vector $\bar{y}$;

(ii) Solve the matching problem (3.7) obtaining the solution x.

Problem (3.9) is used to get a min-square approximate solution $\bar{y}$ to the system of linear equalities (3.8). It is interesting to note that (3.9) is a quadratic shortest path problem: in fact, as already noted, the constraints define r path problems on disjoint graphs. Unfortunately, due to the nonlinearity of the objective function, the problem is not decomposable into r disjoint shortest path problems. Although to solve it to optimality can be very costly, a reasonably good solution can be obtained at very low cost by exploiting the following consideration: If all y^k vectors but one are held fixed, the resulting problem is a shortest path problem; this is due to the fact that y^k is a binary vector. Then (3.9) is solved iteratively, one shortest path at time, starting from $k = 1$ to $k = r$, and repeating the complete cycle until no improvement is obtained. In [13] the solution is further improved by means of a neighborhood search.

In deriving (3.7), constraint $x \in X$ has been disregarded. To take into consideration such a constraint, a lagrangean relaxation technique can be used as in [7] or, within a different framework, in [15] as illustrated in the next section.

3.5. *Lagrangean relaxation*

Let us modify problem (3.3) by relaxing inequality constraints (ii) into inequalities of the $\geqslant$ type. This change is made to assure feasibility without explicitly inserting the variables corresponding to trippers; the role of such variables is now implicitly taken by the slack variables. Since usually the set X is defined by a (very) small number of linear inequalities, we rewrite constraints (v) as follows:

$$Bx \geqslant a, \tag{3.10}$$

where matrix B and vector a have conformable dimensions.

Consider now the lagrangean relaxation of (3.3) with the above introduced changes:

$$\phi(\lambda, \mu) = \min cx - \sum_k \sum_p \lambda_{pk} \left(\sum_{j \in I_{pk}} x_j - y_p^k \right) - \mu(Bx - a),$$

s.t.

$$E^k y^k = b^k, \quad k = 1,\dots,r, \tag{3.11}$$

$$y^k \in \{0, 1\}^{m_k},$$

$$x \in \{0, 1\}^{|D|},$$

where λ and μ are non negative multiplier vectors. Relaxations of this type are described in [15,33].

The function $\phi(\lambda, \mu)$ is concave piecewise linear, and provides a lower bound to the optimal value of (3.3). The best lower bound can be obtained by solving the generalized dual:

$$\operatorname*{Max}_{\lambda,\mu \geqslant 0} \phi(\lambda, \mu), \tag{3.12}$$

which can be done by means of subgradient optimization techniques.

Let $(\bar{x}, \bar{y})$ be a solution of (3.12), then the vector $\bar{y}$ corresponds to a set $\bar{P}$ of pieces of work which satisfy the service requirement on all the blocks. The same is not true for $\bar{x}$, which, at least in general, does not correspond to a feasible schedule. Thus the problem is to match together the pieces of work of $\bar{P}$ in order to get a set of feasible duties; this can be done by solving a matching problem as illustrated in the previous section.

4. Rostering

Once a feasible crew schedule has been obtained, a further problem still remains to be solved, the *rostering problem*, i.e. to assign the duties to the actual drivers. Depending on the union contract clauses, this can be either a trivial problem or a rather difficult one. In most North American Companies the assignment of the duties is left to the drivers themselves and the decision is made on a seniority basis; that leaves little or no space at all for mathematical modeling. On the other hand there are cases in which the union contract is so binding that, in order to computerize the rostering procedures, one has to resort to rather sophisticated mathematical techniques. This is the case with many European Companies, where, other things being equal, the objective is set to distribute the work load evenly among the drivers. We shall address ourselves to this last type of problem and will show how network models can be helpful in this context too.

4.1. *The multilevel assignment model*

We describe now a rather simplified model of the rostering problem, then we will show how this model can be used in more realistic cases.

Let m time intervals (days, weeks, etc.,...) be given. For each time interval k ($k = 1,\dots,m$), n tasks are defined each with a non-negative weight w_{ki} ($i = 1,\dots,n$). We consider the problem of assigning the tasks to n workers in such a way that:

(i) each worker is assigned one task for each time interval:

(ii) the total work load (sum of the weights) is evenly distributed among the workers;

(iii) some simple feasibility constraints are satisfied, which prevent some pairs of tasks from being assigned in consecutive time intervals to the same worker.

The problem can be phrased as a network one [14]. Introduce the m-partite graph $(N_1,\dots, N_m, A_1,\dots,A_{m-1})$ where the set of nodes N_k, $k = 1,\dots,m$, contains one node for each task to be performed in the kth time interval, while the set A_k ($k = 1,\dots,m-1$) contains arcs from nodes of N_k to nodes of N_{k+1}, corresponding to those pairs of tasks that can be assigned consecutively to the same worker. A feasible solution to this problem corresponds in the graph to a set of n disjoint paths from the nodes of N_1 to the nodes of N_m. The nodes of each path correspond to the tasks assigned to one single worker; the sum of their weights is the work load for that worker. The problem can also be viewed as a multilevel assignment; in fact, for $k = 1,\ldots,m-1$, the arcs in the paths connecting the nodes of N_k to the nodes of N_{k+1} define an assignment in the bipartite subgraph (N_k, N_{k+1}, A_k).

Out of the many feasible solutions we look for one in which the work load for each worker is as close as possible to the average work load $(1/n)\Sigma_{i,k}w_{ki}$. This goal can be implemented by means of different objective functions, depending on which 'norm' one wants to use to measure the distance between two vectors. If the norm L_∞ is used, the problem becomes to find one solution out of those feasible, such that the largest work load is minimized. This objective function leads to a *Multilevel Bottleneck Assignment* problem; although NP-hard, this problem can be heuristically solved at low computational cost yielding quite good suboptimal solutions [14].

4.2. Rostering by decomposition

The actual rostering problem is far more intricate than the simple problem we have considered so far. Usually three sets of duties are given: D_s contains the Saturday duties, D_h the holiday and Sunday duties and D_w the weekday duties. Each driver must be allowed q days off after p consecutive regular workdays. A lower bound is set on the difference between ending and starting times of consecutive duties assigned to the same driver. In some cases, as an additional requirement, the first duty after the off period has to be a late duty while the last one must be an early duty.

If, for the sake of simplicity, we consider $q = 1$ and $p = 5$, as is most often the case, then an assignment of duties to a driver consists of a sequence of *quintuplets* of duties with days off in between on the basis of the *work assignment matrix* described in Table 5 which provides the work assignment for a single driver for 6 weeks. The stars denote days off, while d_{ij} ($i = 1,\ldots,7$, $j = 1,\ldots,5$) is the jth duty of the ith quintuplet. It is easy to see that the same matrix can be used for the weeks from the 7th to the 12th, and so on to the end of the planning horizon.

To build up a work assignment matrix, seven different types of quintuplets are needed: first one quintuplet with a holiday duty followed by four weekday duties, then one quintuplet starting with a saturday duty followed by a holiday duty and three weekday duties and so on; the seventh is a quintuplet with only weekday duties.

Table 5

Weeks	Days						
	Sa	Su	Mo	Tu	We	Th	Fr
1	*	d_{11}	d_{12}	d_{13}	d_{14}	d_{15}	*
2	d_{21}	d_{22}	d_{23}	d_{24}	d_{25}	*	d_{31}
3	d_{32}	d_{33}	d_{34}	d_{35}	*	d_{41}	d_{42}
4	d_{43}	d_{44}	d_{45}	*	d_{51}	d_{52}	d_{53}
5	d_{54}	d_{55}	*	d_{61}	d_{62}	d_{63}	d_{64}
6	d_{65}	*	d_{71}	d_{72}	d_{73}	d_{74}	d_{75}

Without going into details, we state next the main steps of the decomposition approach used in [12] to solve the rostering problem:

1. All the quintuplets of each type needed to cover the daily service are generated. This is done by solving seven multilevel bottleneck assignment problems. To set up these problems a pre-processing phase is necessary in which the sets D_s, D_h and D_w are partitioned into 5 subsets each: the jth subset of each duty set contains duties to be used in jth position in the quintuplets. Dummy duties are added to make the cardinalities of all the subsets equal. These dummy duties will be useful at a later date in the daily operation of the system: because the drivers with dummy duties in the roster are eligible to replace absent ones.

2. From the quintuplets a set of work assignment matrices is constructed. If appropriate, more matrices are packed together to generate work assignments for each driver for the whole planning horizon. These operations too are performed by solving multilevel bottleneck assignment problems.

The results obtained with this approach in practical applications have proved to be very successful.

References

[1] E. Baker, L. Bodin, W. Finnegan and R. Ponder, Efficient heuristic solution to an airline crew scheduling problem, *AIIE Trans.*, *12* (2) (1979) 79-85.

[2] E. Baker, L. Bodin and M. Fisher, The development and implementation of a heuristic set covering based system

for air crew scheduling, Working Paper #80-015, University of Maryland (1980).

[3] E. Balas and M. Padberg, Set partitioning: A survey, *SIAM Rev. 18* (4) (1976) 710–760.

[4] E. Balas and A. Ho, Set covering algorithms using cutting planes, heuristics and subgradient optimization: A computational study, *Math. Programming Study 12* (1980) 37–60.

[5] M. Ball, L. Bodin and R. Dial, Experimentation with a computerized system for scheduling mass transit vehicles and crews, in: A. Wren, Ed., *Computer Scheduling of Public Transport* (North-Holland, Amsterdam, 1981).

[6] M. Ball, L. Bodin and R. Dial, A matching based heuristic for scheduling mass transit crews and vehicles, Working Paper #80-007, University of Maryland (1981).

[7] M. Ball, L. Bodin and J. Greenberg, An enhancement to the Rucus2 crew scheduling system, *3rd International Workshop on Transit Vehicle and Crew Scheduling*, Montreal (1983).

[8] M. Ball, L. Bodin and R. Dial, A matching based heuristic for scheduling mass transit crews and vehicles, *Transportation Sci. 17* (1983) 4–31.

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

[10] L. Bodin, D. Rosenfield and K. Kydes, UCOST: A micro approach to a transit planning problem, *J. Urban Anal. 5* (1) (1978) 47–69.

[11] R.E. Burkard and U. Derigs, *Assignment and Matching Problem: Solution method with FORTRAN Programs* (Springer, Berlin, 1980).

[12] R. Belletti, P. Carraresi, A. Davini and G. Gallo, BDROP: A package for the bus drivers' rostering problem, *3rd International Workshop on Transit Vehicle and Crew Scheduling*, Montreal (1983).

[13] P. Carraresi, G. Gallo and J.M. Rousseau, Relaxation approaches to large scale bus driver scheduling problems, *Transportation Res. 16B* (1982) 383–397.

[14] P. Carraresi and G. Gallo, A multi-level bottleneck assignment approach to the bus drivers' rostering problem, *European J. Operational Res. 16* (1984) 163–173, this issue.

[15] P. Carraresi, A. Davini and G. Gallo, Bus Drivers' scheduling: A Lagrangean relaxation approach, *3rd International Workshop on Transit Vehicle and Crew Scheduling*, Montreal (1983).

[16] P. Carraresi, A. Davini and G. Gallo, A heuristic approach to the multiple depot vehicle scheduling problem, in preparation, to be presented at TIMS/ORSA Annual Meeting, San Francisco (1984).

[17] N. Christofides, *Graph Theory: An Algorithmic Approach* (Academic Press, New York, 1975).

[18] L.R. Ford, Jr. and D.R. Fulkerson, *Flows in Networks* (Princeton University Press, Princeton, NJ, 1962).

[19] U. Derigs, Solving matching problems via shortest path techniques, *NETFLOW 83*, Pisa (1983).

[20] B. Gavish, P. Schweitzer and E. Shlifer, Assigning Buses to schedules in a metropolitan area, *Comput. Operations 5* (1978) 129–138.

[21] J. Hoffstadt, Computerized vehicle and driver scheduling for the Hamburger Hochbahn Aktiengesellschaft, in: A. Wren, Ed., *Computer Scheduling of Public Transport* (North-Holland, Amsterdam, 1981)

[22] J.L. Kennington and R.V. Helgason, *Algorithms for network Programming* (Wiley, New York, 1980).

[23] J.K. Lenstra and A.H.G. Rinnooy Kan, Complexity of vehicle routing and scheduling, *Networks 11* (1981) 221–227.

[24] R. Lessard and J.-M. Rousseau, An enhancement to the Hastus crew scheduling algorithm, *3rd International Workshop on Transit vehicle and Crew Scheduling*, Montreal (1983).

[25] R. Lessard, J.-M. Rousseau and D, Dupuis, Hastus I: A mathematical programming approach to the bus driver scheduling problem, in: A. Wren, Ed., *Computer Scheduling of Public Transport* (North-Holland, Amsterdam, 1981).

[26] R. Marsten, M. Muller and C. Killion, Crew planning at flying tiger: A successful application of integer programming, *Management Sci. 25* (1979) 1175–1183.

[27] R. Marsten and F. Shepardson, Exact solution of crew scheduling problem using the set partitioning model: Recent successful application, *Network 11* (2) (1981) 167–177.

[28] G. Mitra and K. Darby-Dowman, CRU-SCHED: A computer based bus crew scheduling system using integer programming, *3rd International Workshop on Transit Vehicle and Crew Scheduling*, Montreal (1983).

[29] J.B. Orlin, Minimizing the number of vehicles to meet a fixed periodic schedule: An application of periodic posets, *Operations Res. 30* (4) (1982).

[30] C.S. Orloff, Route constrained fleet scheduling, *Transportation sci. 10* (2) (1976).

[31] C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity* (Prentice-Hall, Englewood Cliffs, NJ, 1982).

[32] J.M. Rousseau (Editor), *Proceedings of the 3rd International Workshop on Transit Vehicle and Crew Scheduling*, Montreal (1983).

[33] W. Shepardson and R. Marsten, A Lagrangean relaxation algorithm for the two duty period scheduling problem, *Management Sci. 26* (1980) 274–281.

[34] A. Wren (Editor), *Computer Scheduling of Public Transport* (North-Holland, Amsterdam, 1981).