PERGAMON

Computers & Operations Research 27 (2000) 1249–1269

computers & operations research

www.elsevier.com/locate/dsw

A comparison of different solution approaches to the vehicle scheduling problem in a practical case

F. Baita[a], R. Pesenti[a], W. Ukovich[b,*], D. Favaretto[c]

[a]*Dipartimento di Elettrotecnica, Elettronica ed Informatica, Università di Trieste, Italy*
[b]*Dipartimento di Automatica ed Informatica, Università di Palermo, Italy*
[c]*Dipartimento di Matematica Applicata ed Informatica, Università di Venezia, Italy*

Received 1 August 1996; received in revised form 1 August 1998

Abstract

The Vehicle Scheduling Problem (VSP) consists in assigning a set of scheduled trips to a set of vehicles, satisfying a set of constraints and optimizing an objective function. A wide literature exists for the VSP, but usually not all the practical requirements of the real cases are taken into account. In the present paper a practical case is studied, and for it a traditional method is tailored and two innovative heuristics are developed. As the problem presents a multicriteria nature, each of the three algorithms adopts a different approach to multicriteria optimization. Scalarization of the different criteria is performed by the traditional algorithm. A lexicographic approach is followed by an algorithm based on logic programming. Finally, a Pareto optimization approach is implemented by a modified genetic algorithm. All the algorithms are tested on the real problem, and two of them produce interesting practical results.

Scope and purpose

This paper presents the practical experience with a real case of Vehicle Scheduling Problem (VSP). The VSP is a classical optimization problem which is faced in the operational planning of public transportation systems (see for instance Dantzig and Fulkerson (Naval Research Logistics Quarterly 1954;1:217–222)). It consists in assigning a set of scheduled trips to a set of available vehicles, in such a way that each trip is associated to one vehicle and a cost function is minimized. For some versions of it, such as when all vehicles are equal and share the same depot, efficient algorithms exist (see for instance Bodin et al. (Computers & Operations Research 1983;10:63–212), Carraresi and Gallo. (European Journal of Operational Research 1984;16:139–151)); nevertheless, real-life applications often turn out to be complex, due to the particular requirements which are present in practical situations, but are hard to be modeled.

Practical requirements for this problem, usually not considered in the literature, include considering several criteria, producing different alternative solutions, and getting hints on how data could be modified to improve the effectiveness of the solutions.

* Corresponding author. Tel.: + 39-40-676-3410; fax: + 39-40-676-3460.
E-mail address: ukovich@univ.trieste.it (W. Ukovich)

PII: S0305-0548(99)00073-8

The paper analyzes the features of the real problem and discusses different algorithmic approaches for it. It has basically two purposes. The first is to analyze, formalize and comply with the experienced requirements of the practical problem. The second consists in assessing the applicability and performance of non-conventional heuristics and of a traditional exact method.

Keywords: Vehicle scheduling problem; Multicriteria optimization; Assignment problem; Logic programming; Genetic algorithms

1. Introduction

This paper presents the analysis of a practical case of Vehicle Scheduling Problem (VSP) concerning the urban public transportation system of the city of Mestre (Venice, Italy). The case study has been proposed by the Public Transportation Company of Venice.

The VSP is a classical optimization problem. There are efficient algorithms for some versions of the problem, i.e., when all vehicles are equal and share the same depot (see for instance [1,2]). Nevertheless, real-life applications may turn out to be complex due not only to the dimension of the problem but also, and more important, to the particular requirements which are present in practical situations but are hard to be modeled. Practical requirements for this problem, usually not considered in the literature, include considering several performance evaluation criteria, producing different alternative solutions, and getting hints on how data could be modified to improve the quality of the solutions.

According to the above considerations, this study has two purposes: to analyze, formalize and comply with the experienced requirements, and to assess the applicability and performance of non-conventional heuristics and of a traditional exact method.

The survey by Bodin et al. [1] appears to be the most comprehensive treatment of the whole field of routing and scheduling problems. The authors describe the transformation of the VSP in an easy, well-known optimization problem: the minimum cost flow problem. This formulation has been first devised by Dantzig and Fulkerson [3], and is thoroughly reviewed by Carraresi and Gallo [2].

A recent overview by Daduna et al. [4] shows the state of the research in different vehicle scheduling problems: the basic VSP, the VSP with multiple depots, the VSP with different vehicle types, and the VSP with maximum time constraint.

More in general, Odoni et al. [5] present a review of Operations Research techniques in public urban transit problems. In particular, they point out that there are many relations between the different elements of the transit system, from the design of the infrastructures to the schedules of vehicles and crews. It is not viable, however, to take into account all the elements together: the whole problem has to be split into tractable pieces. While solving each single piece, though, the interactions with the other components of the system have to be taken into account.

Dell'Amico [6] formulates the VSP as a particular transportation problem and describes a new efficient algorithm based on a shortest path search; it is applied to randomly generated test problems.

Bertossi et al. [7] show how the VSP can be reduced to a minimum cost perfect matching problem in a bipartite graph, also called Assignment Problem (AP). Varying the cost parameters, the problem turns either to a *minimum fleet size*, or a *minimum operational cost* problem (which minimizes deadheading trips and idle times), or a combination of both. For this algorithm, no computational results are reported.

Carpaneto et al. [8] analyze the most efficient algorithms for the Assignment Problem. The authors claim that even if the Assignment Problem can be solved through any Linear Programming algorithm, the ones they consider are more efficient in different classes of randomly generated test problems.

Paixão and Branco [9] present another similar algorithm, the *quasi-assignment* algorithm, which minimizes the operating costs when the number of vehicles is fixed. Basically, it is a modification of the Hungarian algorithm for the Assignment Problem (cf., for instance [10]); it is also extended to the multiple depot case. This algorithm is applied to some real-life applications at Rodoviária Nacional, the largest road mass transport operator in Portugal.

Many commercially viable software packages, which perform the scheduling by integer linear programming methods, are also described in the literature, in particular in the proceedings of the most recent workshops on computer scheduling of public transport (see [4,11]).

To our best knowledge, all the requirements of our case (see Section 2) have never been considered in the existing literature; furthermore, the VSP never seems to have been dealt with by methods like the Genetic Algorithms and the Logic Programming, as the ones described in the following sections.

Actually, modeling the real problem faced in this paper as a VSP is not completely satisfactory. In fact, it also involves aspects that are quite difficult to formalize. Furthermore, it turned clear after some first attempts that it was unrealistic to assume that all the "knowledge" on the real problem (in terms of both the problem formulation and problem data) could be some time completely available. Actually, the solution of the VSP is only one operation in a more complex dynamic decisional process, which also includes negotiation among human actors. Then, both objectives and constraints were not clearly defined and changed through the time.

For the above reasons, a natural choice was to face the problem by developing a flexible Decision Support System (DSS) with robust capabilities for automatic rescheduling [12]. As suggested in [13], Artificial Intelligence tools and languages seemed appropriate to model and represent such knowledge. In addition, literature reports many DSSs developed for scheduling tasks in a real environment: since the 30 systems reviewed in [14] in 1988, many others have been developed.

Although some significant reports of unsuccessful implementations exist (e.g., [15]), literature usually present stories of successful implementations with, however, few hints about the utilization of the system in the real case. Therefore, this research has two main purposes. The first is to compare a classical approach with limited modeling capabilities with – apparently – more flexible approaches such as Logic Programming and Genetic Algorithms. The second is to present some lessons learned in developing a system for a real-world application.

The implemented system significantly differed from the initially conceived one, since the problem domain became more and more complex as the system was being developed. On the other hand, the limits of human experts in dealing with an automated decisional process turned out to be a crucial factor. Conclusions of this type agree with the results discussed in [15].

The paper is organized as follows. First, the most relevant characters of the real problem of interest are discussed in Section 2. Then in Section 3 the general concepts used to approach the considered problem are outlined. The three methods that have been implemented, i.e., the Assignment Model, the Logic Programming, and the Genetic Algoritms, are presented in Sections 4, 5, and 6, respectively. The results they provide are then compared in Section 7. Some concluding remarks are eventually sketched in Section 8.

2. Problem description

The VSP consists of assigning vehicles to time tabled trips in such a way that each trip is carried out by one vehicle, a set of constraints is satisfied, and a cost function is minimized. The problem data are a set of trips, each one defined by the starting and ending times and locations. The objective consists of minimizing the number of vehicles and different operational costs. These costs are often related to deadheading trips (gas, driver, etc.) and to useless idle times.

In general, the constraints may refer to different elements, such as the length of either time or distance during which a vehicle may stay on duty before returning to the depot for servicing or refueling, or the restriction that certain tasks can only be performed by certain vehicle types, or also the presence of multiple depots where vehicles may be housed. In the case we consider these kinds of constraints are not present.

2.1. Real problem features

The VSP related to the urban bus service of Mestre shows some features that may be typical in several situations of mass transit scheduling problems, although not usually considered in the literature. They are discussed in this section, after the main characteristics of the system are sketched.

The ordinary daily duty consists of more than 2500 trips, serving 80 locations, divided in 46 lines and performed by over 200 buses.

Buses cannot be assigned to any line: there are three classes of vehicles of different length: "long", "regular" and "short" buses. The set of trips to be performed by vehicles of different classes are disjoint: the whole problem is divided in three separate subproblems. Furthermore, the short bus trip set contains very few trips (66), which present such a structure that makes it very easy to schedule them. Therefore, the actual objects of the study are the subproblems related to the first two classes of vehicles.

2.1.1. Multicriteria nature of the problem

In practice, it turns out that there are several criteria for assessing a solution. The most relevant ones, which have been proposed by the company managers, are the following.

- Minimizing the number of buses. If this is the main objective, then the problem turns out to the so-called *minimum fleet size problem* (cf. [7]). The number of vehicles strongly affects the strategic investment planning, because of the very high capital cost of the buses. In the present case the actual fleet size is given but there are problems related to the bus servicing: an effective

scheduling would make it easier managing resources. As a consequence, reducing the number of required buses turned out to be a sensible requirement.

- Minimizing the number of line changes. Even if there is no binding constraint for a vehicle to stay on the same route all day long, it is advisable to minimize the number of line changes, for drivers convenience. Actually, a schedule with several changes is not appreciated by the personnel and a vehicle scheduling that takes it into account makes easier the crew scheduling problem.
- Minimizing the number and length of deadheading trips. Deadheading trips generate some operational costs related to fuel and drivers time. Furthermore, customers do not like watching empty buses running while they are waiting at a stop. Also, the local government financial support to such a public company is usually proportional only to the regular trips.
- Minimizing the idle times of the vehicles at the termini. Extra idle times at the termini may increase the personnel cost or make more difficult the crew scheduling problem (this is another example of the relation between the different phases of the whole problem of the transit system planning, as pointed out by Odoni et al. [5]).

These criteria are often conflicting. Furthermore, some of them may have different impacts and relevance in practice. It is obvious that, for instance, using one additional vehicle to save one deadheading trip or one line change is not a good choice, but it is not easy to establish a clear hierarchy or a quantitative tradeoff between these criteria. For example, the exact amount of savings in terms of line changes and deadheading trips that can justify one more vehicle can be difficult to assess a priori. In fact, the issue is even more complex, since the relative importance of these criteria is not always clearly stated: actually, it is perceived to be time dependent, in the sense that it changes in different time intervals. There is a peak period, in our case from 6.30 to 8.30 a.m. The whole amount of needed vehicles is determined in this peak interval, during which it is sensible that the criterion of minimizing the number of vehicles takes an higher priority.

These features make it useful to devise a method that yields a set of different solutions that can later be evaluated by the decision maker. In this way, the awkward problem for the analyst of deciding a priori a rigid preference structure for the objective functions is converted to a simpler a posteriori assessment of some alternative solutions by the decision maker.

2.1.2. Further requirements

In addition to the usual constraints regarding the feasibility of the schedule, two further requirements have been proposed: the maximal idle time allowed at the termini and the "double trip constraint". According to the company managers, if the idle time at the terminus is over a given threshold, then the vehicle must come back to the depot and later return to the terminus to continue its duty. This constraint is, again, related to crew scheduling requirements, in order to avoid useless idle time.

The "double trip constraint" consists of a particular requirement of the crew scheduling problem associated with the structure of the trips and idle times between them. According to a trade union agreement, if there are two following trips both over 25 min long, it is necessary to grant the driver 20 min of overall pause, which can be split in the three pieces: before the first trip, between the first and the second, and after the second. It is straightforward that a vehicle scheduling that takes into account this requirement can strongly facilitate the crew scheduling problem. Requirements like

this one are difficult to model and in general to manage; for this reason, it is convenient for company managers to have many feasible solutions for the same instance of the problem, as it was pointed out above.

2.1.3. Data of the problem

The input data for the VSP are the time tabled trips and the transfer times between termini. The timetable is built in function of the density of the demand: depending on the particular route (line) and time of the day, the frequency of the trips is defined. A good level of regularity is also necessary in the timetable, for users' convenience, but the company managers claim that in some cases it is possible to modify the schedule of some trips, provided that this implies an improvement in terms of some of the considered criteria.

However, modifying the schedule of a single trip has to be done with care because shifting (i.e., delaying) one trip may affect "on cascade" all the subsequent trips of the same schedule, breaking its feasibility. Furthermore, such modifications are not allowed for all the trips, for instance due to a possible connection between trips of different schedules at the same terminus. Again, a quantitative tradeoff between the amount of the time shifts and the possible related benefits cannot be easily defined. Again, all these considerations enforce the need for multiple solutions in order to evaluate ex post, on a qualitative basis, what is not easy to be modeled ex ante.

The availability of the transfer time values between termini is another problematic issue: they are not measured by the company, because they are not relevant for their plans which are often based on programmers' experience. A structured optimization method, instead, needs all these values, or at least their estimates.

3. Solution of the problem – general approach

In this section we discuss the approaches we have adopted for the problem presented in Section 2. Some of the requirements have been addressed by appropriately formulating the models or the data for them, other ones by algorithmic elements.

3.1. Data and modeling issues

All the implemented methods have been tested on a data set provided by the company, corresponding to an actually implemented timetable. The obtained solutions have been compared with the situation in use. When the actual transfer time between termini was not known, we considered a standard transfer time of 20 min: in most cases it is a conservative estimate. This is an a priori assumption whose effectiveness needs to be checked a posteriori by the decision maker: the value can be obtained by means of an iterative process of hypothesis/verification. It bears the risk of long iterations, but also the advantage that only a subset of data is actually required.

In the models of the literature a short-time interval is often forced between two subsequent trips to face a possible delay. In our applications, we did not insert it, according to the current use of the company. This fact may be justified by the hypothesis that the route time implicitly includes this interval. Furthermore, we analyzed the possibility of inserting a negative interval between two subsequent trips performed by the same vehicle. This means that the second trip can start before

(some minutes at most) the end of the first trip. Obviously, this implies that the starting or ending times of one of the two trips are modified, to make feasible the solution: such shifts are acceptable, as explained above. Moreover, it is necessary to take into account at least two conditions. One is that the feasibility of the remaining part of the schedule has to be maintained. The other one is that the number of negative intervals has to be low, to avoid to distort the timetable too much. The latter condition can be satisfied by penalizing the violations of the trip times in the objective function, or inserting a new constraint.

In this way, it is possible to obtain more satisfactory solutions, simply by modifying the times of some trips. In other terms, the tool for solving the VSP can give also an informative feedback for the construction of an appropriate trip schedule. The possibility of timetable shifts may be found in some commercial software packages, which point out the need of this sort of sensitivity analysis of the timetable. In those cases, however, a separate sensitivity analysis function module needs to be implemented. It can be applied after the schedule optimization module [16], or it can interact with it [17], but it is not inside the optimization module, as it is in our model using negative slack times.

3.2. System implementation history

In this section, the steps gone through by the authors in developing the DSS are briefly sketched. They are the results of several discussions between the system developers and both the experts and the management of the transit authority.

1. first, a prototypal system based on a Logic Programming approach to the problem has been developed. As described in Section 5, the initial results seemed promising;
2. in developing a more advanced prototype of the system, difficulties arose in the acquisition of new knowledge. In particular, it was necessary the presence in the transit company of a specific professional figure to formulate the new Prolog statements. This solution turned out unacceptable for the management. In addition, each new statement was more and more specific, i.e., fitted fewer and fewer situations. With the new expanded knowledge base, the system became slow and the quality of the provided solutions did not increase proportionally;
3. it appeared sensible to produce some basic solutions, obtained by modifying as much as possible the classical VSP formulation but, at the same time, maintaining tractable the problem (see Section 4). The initial solutions may be used to perform a posteriori local searches and to evaluate the quality of the obtained solutions. It did not turn out of significant help to try to precisely tune the weights used in the objective function considered. On the contrary, users seemed more interested in the possibility of assessing different tentative values for the different cost components and possibly to correct by hand the solutions by means of a graphical interface;
4. to improve the a posteriori local searches, a Genetic approach was eventually tested. However, although a new coding for genes was introduced, the improvements of the obtained results turned out to be not competitive with the corrections that could be introduced by a human expert in the same time.

Overall, the users appeared satisfied of a system quickly producing some tentative solutions, and allowing to correct by hand possible inconsistencies. The "intelligent" part of the solution system (i.e., the Logic Programming-based methods) turned out interesting both in a preprocessing phase,

when it could eliminate a priori some trip concatenations which uselessly lengthen computation times, and in a postprocessing phase, when it may point out the attention of the human expert only to nontrivial trip concatenations. Possible examples of trivial trip concatenations are: successive trips along the same line, or a trip departing from the last stop of the arriving trip.

3.3. Multicriteria issues

The problem features make it sensible to adopt a multicriteria method. Traditionally, multicriteria approaches are basically three (see for example [18]).

- Scalarization: A scalar objective function is obtained as a weighted sum of the different objectives.
- Lexicographic approach: The objectives are implemented as constraints, and feasible solutions are searched relaxing progressively the constraints that represent the objectives with lower priority.
- Pareto optimality: The purpose is finding the Pareto optimal frontier, which is the set of solutions that dominate the other ones, in the sense that they are not worse with respect to every objective, and strictly better for at least one objective.

Each of the three methods we have tested for solving the VSP follows one of these approaches: scalarization is used in the Assignment model to obtain a scalar objective function, the search strategy of the Logic Programming is lexicographic, and the multicriteria Genetic Algorithm searches the Pareto optimal frontier.

4. Assignment model

The VSP may be formulated as an Assignment Problem (AP), which is one of the most studied problems in combinatorial optimization: see for instance [1,2,8]. The basic procedure that we have used and adapted to our problem is described by Carpaneto et al. [8]. They claim that the algorithm they propose is the most efficient among the existing ones; it is based on the well-known Hungarian method. From the computational complexity point of view, it has the merit of being polynomial (while the heuristics considered in the following sections are not polynomial, at least in principle, i.e., in the worst case).

In the Assignment model, all the criteria for the VSP are represented by costs associated to the arcs of the bipartite graph. Two trips performed in sequence by the same vehicle are represented by an arc from the ending node of the first trip to the starting node of the second trip. The arc set A can be divided in two subsets A_1 and A_2:

$$A = A_1 \cup A_2, \tag{1}$$

where A_1 contains the arcs connecting compatible trips, and A_2 contains arcs between trips that are not compatible, and indicate the last trip of the duty of a vehicle and the first trip of the duty of another vehicle. The arcs of A_2 are strongly related to the use the vehicles, so the *capital* costs (those affecting the number of vehicles used) correspond to them. The unit capital cost (i.e., the cost of using one vehicle) is denoted as *nbus*. The arcs of A_1 correspond to the different *operational* costs,

which depend on the decision of coupling two trips so that they are performed in sequence by the same bus.

4.1. Cost structure

Two different types of cost are considered in the AP model: the first heavily affects the size of the fleet, while the second includes the different costs related to the other criteria described in Section 2.1.1. It is clear that if only capital costs are contemplated, for instance by setting

$$c_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in A_2, \\ 0 & \text{if } (i,j) \in A_1, \end{cases}$$

then the problem becomes a minimum fleet size problem, which provides the schedule with the minimum number of buses without considering any other criterion. In our application, however, the construction of the vehicle duties takes into account different possible costs of sequencing trips. A vehicle schedule violating some requirements of the problem is penalized by augmenting the cost of the corresponding arc in A_1 by an amount which depends upon the nature and number of the violated criteria. These objectives are the minimization of deadheading trips, line changes, idle times at the termini, and a number of situations when a vehicle has to stay at the terminus more than a fixed threshold time.

More in detail, operational costs have been decomposed in several elements, which are discussed in the following. While operational costs can be modeled in several different ways, also depending on the specific situation considered, it seems interesting to show in some detail the cost model used in the specific considered case, not only to provide one possible way of facing that situation, but mainly to suggest the flexibility of the proposed approach concerning such an issue.

Three elements affecting operational costs have been considered: deadheading from a terminus to a different one, line changes and waiting times at a terminus.

Deadheading costs have been modeled considering three terms:
dhfix: a fixed cost incurred every time a bus makes a deadheading trip from a terminus to a different one;
dhdist × *distance*: a variable cost, proportional to the *distance* travelled in the deadheading trip;
dhund: an additional fixed cost affecting deadheading trips that are particularly awkward, or undesired, according to the managers' opinion and experience.
For line changes, only a fixed cost *lc* has been contemplated. For the waiting time at a terminus, a more detailed cost structure has been considered. Basically, there is a penalty $wait \times wt$ proportional to the waiting time *wt*, when it lies between prescribed limits *wl* and *wu*:

$$wait \times wt \quad \text{for } wl \leqslant wt \leqslant wu.$$

If *wt* becomes too large, i.e., if $wt > wu$, then the bus is supposed to go back to the depot before starting for a new trip, as discussed in Section 2.1.2. In this case, a fixed cost *depret* is incurred. Finally, the possibility of having negative values for waiting times, as discussed in Section 3.1, is penalized by a variable cost $-negw \times wt$, proportional to the time *wt* of which the start of the following trip has been delayed in order to recover feasibility. In this case, *wt* is allowed to lie in the range

$$-wmin \leqslant wt \leqslant 0. \tag{2}$$

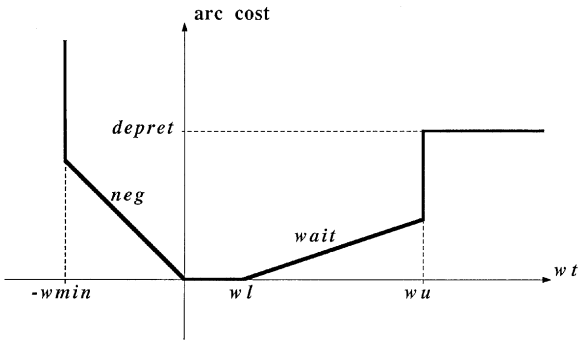

Fig. 1. Cost of an arc as a function of the waiting time *wt*.

The considerations concerning the time criteria for sequencing trips are summarized in Fig. 1, showing the arc cost as a function of the waiting time *wt* between two ordered trips (v_i, v_j).

In the AP model, a scalarization of the criteria is performed, depending on the relative weight of the different types of cost, which are typically fixed by a "trial and error" approach. Given the cost structure (i.e., the weights of the different criteria), the AP method, in fact, gives an exact but (in general) unique solution. It is possible to obtain many different solutions with multiple runs, by modifying the weight structure. The trial and error process of defining the weight structure may also lead to formalize and quantify the relative importance of some criteria affecting the personnel organization. This can be a useful informative tool, for example in the trade union negotiations.

4.2. First computational results

The algorithm has been first implemented in ANSI C language, and the runs were executed on a workstation DEC Alpha 3000/Axp 300 with operating system UNIX OSF/1.

Several solutions have been provided, with different weights for the cost factors considered in Section 4.1. As an example, the figures of a set of solutions are shown in Table 1. They have been obtained with cost factors fixed to the values shown in Table 2. In Table 1, different solutions are

Table 1
Results of the AP algorithm

	Solution 1	Solution 2	Solution 3
Negative slack time (min)	0	3	5
Vehicles	189	182	175
Deadheading trips	374	363	367
Line changes	310	310	329
Trip shifts	/	20 (0.8%)	29 (1.1%)

Table 2
Values of cost parameters for the solutions of Table 1

nbus ($)	*dhfix* ($)	*dhdist* ($/km)	*dhund* ($)	*lc* ($)	*depret* ($)	*wait* ($/min)	*negw* ($/min)	*wl* (min)	*wu* (min)	*wmin* (min)
20,000	1000	20	100	800	2000	100	2000	5	60	5

shown, corresponding to different values: 0, 3′, 5′ for the parameter *wmin* of Eq. (2). As discussed in Section 3.1, allowing negative waiting times requires to change the schedule of some trips to make feasible the solution. The number of trips actually changed is also shown in Table 1, and it is a small fraction of the number of trips.

The algorithm is pretty efficient: solving an instance for the largest problem took little more than one minute of CPU time.

4.2.1. Robustness study

A detailed sensitivity analysis of the solutions considered in the previous section has also been carried out, in order to show how their robustness with respect to variations in the weights of the objective function could be assessed.

First, the tradeoffs between the capital cost (i.e., the number of used buses) and the component of the operational cost given by the number of deadheadings has been experimentally studied. To this aim, several runs of the algorithm have been performed, with different values for the ratio r between the cost factor *nbus* associated to arcs in A_2 as in Eq. (1) and the cost factor *dhfix* introduced in Section 4.1:

$$r = \frac{nbus}{dhfix}.$$

The number of used buses and of the required deadheading trips as functions of r is shown in Table 3 for the peak period of the morning.

Then the effect has been evaluated of modifying each of the values of Table 3. For example, consider *negw*, i.e., the penalty for delaying of 1 min the start time of one trip. Several runs of the algorithm have been performed, with different values for *negw*: 1000, 2000, 5000, 10,000 and 20,000. Table 4 shows the values obtained for the relevant performance evaluators obtained with $wmin = 5'$ and different values for *negw*. In this case, it turns out that values $negw > 10{,}000$ are irrelevant, in the sense that they do not modify the performance of the obtained solution. Similar discussions could be produced for other parameters.

Finally, another assumption has been relaxed: the one of considering separately "long" and "regular" buses (cf. Section 2.1). It has been assumed that buses of one type could also be used on trips reserved to the other type, although with some penalty. The cost *othtype* of using a bus of the "wrong" type on a trip has been considered for different values: 5000, 10,000, 15,000, 17,500 and 20,000. The corresponding performance evaluators provided by the algorithm are shown in Table 5 with $wmin = 0$ for the trips of the rush period: 6.00 ÷ 8.30 a.m.

Table 3
Used buses and required deadheading trips as functions of the ratio r between the cost factors *nbus* and *dhfix*

$r = nbus/dhfix$	0.5	1	1.5	2	2.5	3	3.5	4
No. of used buses	301	298	203	203	190	190	189	189
No. of deadh. trips	0	3	98	98	124	124	127	127

Table 4
Sensitivity analysis for different values of *negw*

negw	1000	2000	5000	10,000	20,000
Used buses	173	175	181	185	185
Deadh. trips	367	367	363	361	361
Line changes	329	329	317	309	309
Trip shifts	43	29	9	0	0

Table 5
Sensitivity analysis for different values of the cost *othtype* of using a bus of the "wrong" type

othtype	5000	10,000	15,000	17,500	20,000
Used buses	180	181	181	184	189
Deadh. trips	81	81	81	75	71
Line changes	97	94	94	83	76
Trips with "wrong" bus type	10	8	8	5	0

In this case, it is worth to point out that while the number of required vehicles increases with *othtype*, the number of deadheading trips and line changes decreases. Moreover, the number of trips with the "wrong" type of bus is quite low. This could suggest a more detailed analysis of such trips since, if "wrong" type buses could be admitted at least for some of them, considerable savings could be achieved in terms of capital costs. This is only a very simple example of the several post optimality arguments that could be discussed using objective scalarization in the AP approach.

4.2.2. Comments

The solutions obtained by the AP approach show a strong improvement on the presently adopted one in terms of used resources; a qualitative analysis of the results points out that most of the requirements of the real problem are also well satisfied. Yet, not all the requirements of the problem can always be met by the Assignment method.

Specifically, according to the company managers there are two kinds of drawbacks with the proposed solutions. The first one consists of some specific points in which the experienced manager's eye can see that a particular trip could be swapped with another one improving the solution in some way. It can be argued that this fact points out that the definition of the exact weight structure is not easy to manage.

The second one is that such a method cannot face the "double trip" requirement. Actually, the assignment method considers only costs affecting the link of two subsequent trips: it cannot handle relationships between three or more trips. This feature may represent a severe limitation of this method. Nevertheless, we choose not to try to formalize the double trip constraint since this would be expected to produce an NP-hard model, thus severely affecting the computational performance

of the algorithm. Rather, we preferred to overcome such modelling limitations in a practical way, by using a relaxed – but efficient – model in order to produce several different solutions; then the decision maker can choose among them.

4.3. Further experiments

The implementation and the results discussed in the previous sections were aimed at providing practical insights for effectiveness issues of the AP approach.

Besides that, it seemed interesting to carry out also an extensive experimentation on random generated data in order to assess the practical performance of the proposed method in terms of the computation times it requires. For this scope, a general purpose package for linear programming (CPLEX – cf. [19]) has been used, in order to get a relatively "implementation free" assessment.

4.3.1. Model for data generation

The simulation experiments have been carried out according to a model stemming from the following empirical observations, which are based on the authors' experience in several cities of Italy:

1. the number of lines is approximately proportional to the square root of the city population;
2. in most cases, lines show a radial pattern, with one terminus near the city center and the other one in a peripheral quartier;
3. trips are concentrated between 5:00 a.m. and 9:00 p.m.;
4. there are three peak hours during the day, around 7:00 a.m., around 12:30, and around 5:00 p.m., with the highest peak at 7:00 a.m. and the lowest at 5:00 p.m. During such periods runs are usually between 50 and 100% more frequent than during the other daylight time periods;
5. the average number of trips per line varies between 40 and 90 depending on the city. Within the same city, there may be lines with 10 trips and lines with more than 120 trips;
6. the average length of a trip is around 20 min, with 15 and 40 min as extreme values.

Assuming the city area approximately proportional to its population, the first observation above leads to the conclusion that the number of lines which cross a unit of area is independent of the overall number of lines of the city. The second observation implies that, for each peripheral terminus, the cumulative probability of finding another terminus within a given distance is proportional to the square of that distance. On the other hand, the same probability is proportional to the inverse of the number of considered lines.

In the simulation experiments, for each city, the number of lines has been defined together with the average number of trips per line. The latter value has been fixed randomly according to a uniform distribution in the interval between 50 and 100.

For each line, the tour length and the number of runs in each direction have been fixed randomly. The former value has been chosen normally distributed with mean 20 min and standard deviation 3 min; the latter value is uniformly distributed on the basis of the city average previously determined, and it falls in an interval whose lower extreme is always 10. The run interdeparture times have been generated according to an exponential distribution with a time-dependent rate shaped to take into account the peak and off peak hours. The same shape has been assumed in all the simulation experiments, since a variability is in any case introduced by the randomness of

Table 6
Solution times for the Assignment formulation

Number of trips	10	100	500	1000	2000	5000	10,000	20,000
Average CPU time (secs.)	0.00	0.08	0.89	2.58	5.78	43.20	259.32	744.06

interdeparture times. To this regard, note that, in order to promote line or terminus changes, neither a constant interdeparture time, nor coordination between the runs in the opposite directions has been assumed.

The termini locations have been decided as follows. First, a number of "transfer centers" has been randomly fixed (according to a Gaussian distribution), with an average of one area every 25 lines. Then, the central terminus of each line has been assigned, with the 66% of probability, to one of such areas. The peripheral terminus has been assumed to belong to one of the main areas only in the 10% of the cases. The time distances between termini not belonging to the main exchange areas have been randomly generated on the basis of the cumulative probability function defined by the second empirical observation above. As an example, when 50 lines are considered, 3 other termini are, on the average, within 15 min from each secondary terminus.

4.3.2. Computational results

The largest test problems faced included 20,000 trips, a larger amount than the number of trips usually carried out by the largest Italian transit companies. Most of the connections between pairs of trips were a priori excluded by inspection. Then, on the average, each trip was involved in 30 possible connections. The average solution times for the different test problems are summarized in Table 6. Runs have been performed on a DEC Alpha 7000/610 workstation, with operating system Digital UNIX v. 4.0B.

Although the experiments with different methods have been carried out on different machines, CPLEX routine turned out to be the most efficient one in solving the VSP, provided it were formulated as a network linear programming problem. Of course, CPLEX is an optimized commercial package, and comparing it with non-professionally implemented prototype programs could not be completely fair. As a consequence, the figures of Table 6 are mostly interesting as they show a typical trend of processing times as a function of the problem size. Nevertheless, since the solution times obtained with CPLEX were acceptable even for large size problems, no effort has been carried out to improve the efficiency of the ad hoc algorithms previously developed for smaller problems. This choice was also justified by the fact that the network simplex algorithm is widely used to solve Assignment Problems [10]. Actually, other algorithms were implemented to guarantee the software portability of the program to solve the real case.

5. Heuristic based on Logic Programming

Logic Programming is a particular class of programming languages (see for instance [20–22]). It seemed interesting to test the capability of this methodology to face the practical requirements of the problem described above. The developed algorithm is a heuristic technique that tries to comply

with the different criteria in a lexicographic way. The criteria are implemented as constraints and the algorithm searches feasible solutions for those constraints, gradually relaxing them when this becomes necessary to build the schedule.

A crucial point to decide is which constraints have to be removed first. The decision is based either on the importance of the corresponding objective or on the variation of the solution space due to the constraint or both. The Logic Programming algorithm for the VSP, in other terms, tries to sequentially link the trips following a priority scheme fixed by the programmer. Defining the priority scheme is an activity similar to defining the weight structure for the assignment model. Among the search strategies that we have tested, a reliable one turned out to be the following:

- first, the algorithm tries to connect the closest trips that are in the same line and same terminus, complying with the possible maximal and minimal idle times at the termini (which are parametrized on the basis of the different intervals of the day);
- if the first search fails, then the constraint of permanence in the same line is removed; trips are linked with those starting from the same location, always meeting maximal and minimal waiting time constraints;
- in the case of a negative result, the search is moved again to other locations, always starting from the nearest one.

The solution tree is scanned with a backtracking technique whenever a connection is achieved, following the same priority order. If even the last criterion fails, then it means that there are no more compatible trips for the current vehicle and the schedule of one vehicle is completed.

By changing the priority scheme, it is possible to get many qualitatively different solutions, thus achieving one of the practical requirements of the problem.

A point of strength of such a method is its flexibility. Additional constraints for the problem can be inserted, allowing to face similar (more difficult) problems, with requirements that cannot be inserted in more traditional methods, as the scalarization of the Assignment model.

5.1. Computational results

Table 7 shows three solutions obtained by a first implementation of a Logic Programming algorithm (cf. point 1 in Section 3.2) with different waiting times at the termini. The results are similar to the ones of the AP method but the algorithm requires less hardware resources (in particular central memory). Using a personal computer with an Intel 486 processor about 2–3 min are required to solve the whole problem. As it was mentioned in Section 3.2, the possibility has then been considered of extending the method to deal with further constraints and requirements, such as the "double trip" one. However, the drawbacks pointed out in Section 3.2, point 2, turned out to be heavier than the corresponding possible benefits.

6. Genetic algorithms

Genetic Algorithms (GA) are general purpose search and optimization techniques inspired by the mechanisms of the evolution of populations. They have been applied to different sectors, dealing with both technical and management problems, showing good performance in terms of

Table 7
Results of the Logic Programming algorithm

	Solution 1	Solution 2	Solution 3
Negative slack time (min)	0	3	5
Vehicles	192	186	175
Line changes	490	500	500
Trips to modify	/	64 (2.5%)	99 (3.9%)

Table 8
Representation of bus schedule for GA

Trip	1	2	3	4	5	6
Vehicle	1	1	2	1	2	2

efficiency/effectiveness tradeoff: see for example the extensive survey by Biethahn and Nissen [23]. For an introduction to the theory of standard Genetic Algorithms, see [24,25].

A modified Genetic Algorithm has been developed for the VSP; it shows some features that make it different from the standard GA [26]. The way a bus schedule is represented is shown in Table 8: each allele (the value inside the string) represents a vehicle, and its position represents the trip assigned to the vehicle. In the example, vehicle 1 is assigned to trips 1, 2 and 4, and vehicle 2 is assigned to trips 3, 5 and 6.

A problem that arises in treating these strings is related to the compatibility among trips assigned to the same vehicle: it might not be maintained after the "cross-over" operator. The following "repairing mechanism" has been adopted: first, the structure of an individual has been characterized by multiple alleles. This means that an individual is represented by a matrix (not by a single vector), where the first column contains the "dominant" alleles. The following columns contain the "redundant" genes that are used to fix incompatibilities, by using a redundant allele instead of the dominant one if the dominant allele implies an incompatibility.

As a first approach, a GA has been applied to the unconstrained VSP with the purpose of finding the minimum number of vehicles that cover all the trips. In this single objective optimization, the "fitness value" f_i of each individual i was calculated as $f_i = 1/(NV_i \cdot NI_i)$, where NV_i is the number of vehicles, and NI_i is the number of incompatibilities of the corresponding solution.

In this way, operational costs are not considered. To face them, a Pareto approach to multicriteria optimization has been used, instead of scalarization (as with AP) or lexicographic order (as with Logic Programming). Each individual is evaluated by means of a vector function, in the sense that the value related to each criterion is explicitly and separately considered.

A central point in the evolution of the population in a GA is how the individuals are selected to be reproduced in the next generation. In order to follow the Pareto approach, two non-conventional selection schemes have been used, in addition to the standard method consisting of assigning to each individual a probability to be selected, which is proportional to the value of the objective function. The first new scheme is a variation of the well-known *tournament selection* [24], based on a set of comparisons between two individuals; the winner is the individual that dominates the other one in the Pareto sense: An individual dominates another one if all its fitness values are not worse, and at least one of them is strictly better. If neither individual dominates the other, one of them is randomly selected. This scheme is similar to the one used by Horn et al. [27], in which the multi attribute optimization is carried on using such a tournament selection scheme, joint with a fitness sharing technique to maintain a high level of variety in the population.

In our approach, the variety is secured by sub-population construction, due to the use of another non-standard selection scheme: the "local geographic selection" for reproduction based on the crossover operator. It is based on the idea that the population has a particular spatial structure: it is divided in *demes* or semi-isolated sub-populations, with relatively thorough gene mixing within the same deme, but restricted gene flow between different demes. The purpose of this scheme is to explore the efficient frontier by not allowing that all individuals evolve towards common characteristics. Tournament selection, which operates among all the sub-populations and assures a (low) mixing of genes between the demes, is maintained as a complement to the local selection.

An extensive experimental analysis has been carried out on the number of individuals, redundant genes, parameter settings and use of the different selection schemes. For a complete discussion of them, see [26].

6.1. Computational results

The algorithm has been implemented in ANSI C language, and the runs were performed on a workstation DEC Alpha 7000 with operating system Open VMS. An advantage of this algorithm is the fact that it intrinsically gives a set of solutions, instead of an unique one: thus one of the practical requirements is achieved in a "natural" way. The liability of this method is its inefficiency. The particular data structure requires long computational times that make it hard to solve real size problems. To face our VSP, it has been necessary to divide the trip set in subsets, identified by the existing different intervals of the day (the peak time interval from 6.30 to 8.30 a.m. and the ones before and after it).

Table 9 shows the results in term of number of vehicles for the three time intervals. These results are practically the same obtained by the other two methods (which are both much more efficient). Even with this partition, a complete solution could take several hours of CPU time.

The application of the multicriteria GA is even more inefficient, so it has been applied only to a test problem: the one related to the second (peak) time interval with long buses. Efficiency questions led to consider two criteria only: minimizing the number of vehicles and deadheading trips.

In Fig. 2 the evolution of the population is showed: the number of vehicles is the same as the corresponding application of the single objective GA, while the number of deadheading trips is reduced (28 vs. 52).

Table 9
Number of vehicles with GA

Time interval	Long vehicles	Regular vehicles	Total
1	30	47	77
2	84	104	188
3	52	58	116

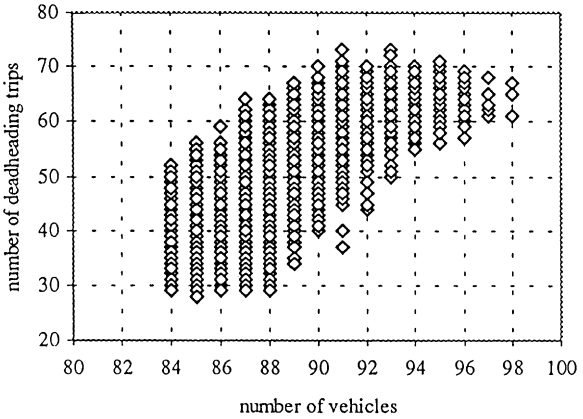

Fig. 2. Representation of the evolution of the population.

These results point out that GA seems not to be at the same level of the other models, and discouraged to try to extend GA search to other criteria, since introducing more criteria implies further raising computation times. However, the method is very flexible and could in principle be extended to difficult problems like the multi depot or other more constrained cases, which are NP-hard problems and cannot be easily solved by traditional methods.

7. Comparison of the results

Since the GA shows a quite poor performance both in terms of efficiency and of effectiveness, as pointed out in Section 6.1, a comparison is considered only between the AP and Logic Programming methods. For the sake of convenience, the results provided by these methods are shown in Table 10. Detailed efficiency figures, such as CPU times, are not reported, since experiments have been carried out on machines with different performance classes.

Comparisons show a clear advantage for the AP method. Its solutions almost always use less vehicles, with less line changes and less trips to modify. Even where more vehicles are required by

Table 10
Comparison between AP and Logic Programming results, with different slack times

Slack time	0 min		− 3 min		− 5 min		—
Method	AP	Logic Pr.	AP	Logic Pr.	AP	Logic Pr.	Real Situation
Vehicles	189	192	186	182	175	175	202
Line changes	310	490	310	500	329	500	420
Trip shifts	/	/	20 (0.8%)	64 (2.5%)	29 (1.1%)	99 (3.9%)	/

the AP solution, the number of line changes and of trip shifts are sensibly reduced with respect to the solution provided by Logic Programming. Furthermore, it should be pointed out that AP allows to keep control of the tradeoff between these elements by fixing the values of the weights for the different considered criteria. Anyway, the Logic Programming algorithm is an interesting tool due to the possibility it provides of extending the scope of search to more constrained problems.

Summing up, GA turns out to be quite inefficient. Logic Programming, instead, seems to be more efficient. However, it does not guarantee optimality (although optimality is achieved in most cases). Therefore, it lacks effectiveness. Although it could account for difficult conditions, this line of development has been abandoned, as it would require skills in Logic Programming that are not available in traditional transit companies. Furthermore, trying to improve its effectiveness results in a dramatic fall of efficiency.

As for computational complexity, AP is polynomial, while Genetic Algorithms and Logic Programming are both exponential in the worst case. However, Logic Programming showed a good performance in our experiments. Finally, AP is a method which can be expected to solve problems of a larger size than could be encountered in practice, as it is shown by the experiments reported in Section 4.3.2.

8. Conclusions

Good solutions for a real case of a Vehicle Scheduling problem have been found, with a significant reduction of the used resources, and an incomparable difference of computational times with respect to the previous way of facing the problem (in the company, several persons work for about one month to get the same result). The solutions have been evaluated by the company's management, and both the AP and Logic Programming methods have been considered worthy of practical use in the programming process, even if not all the requirements can be completely met. In other terms, it is still necessary to review the set of the different solutions to choose the best suitable one, and possibly to introduce some specific manual modifications.

The Logic Programming, which is – like GA – a completely innovative approach for this problem, makes it possible to obtain results qualitatively almost equivalent to the results of the AP method, although it requires non conventional knowledge that can make it harder to be handled by company programmers. The GA, instead, seems to be much more inefficient and unable to obtain the same level of results.

Acknowledgements

This study has been supported by the CNR (the Italian National Research Center) within the PFT2 project under contracts CO94.01346.74 and CO94.01472.74.

We gratefully acknowledge the company management and personnel for their useful contribution. We also gratefully acknowledge Dr. Giovanni Cappelletto for the implementation of the Logic Programming algorithm and Dr. Davide Treu for the implementation of the AP algorithm.

References

[1] Bodin L, Golden B, Assad A, Ball M. Routing and scheduling of vehicles and crew – the state of the art. Computers & Operations Research 1983;10:63–212.

[2] Carraresi P, Gallo G. Network models for vehicle and crew scheduling. European Journal of Operational Research 1984;16:139–51.

[3] Dantzig G, Fulkerson D. Minimising the number of tankers to meet a fixed schedule. Naval Research Logistics Quarterly 1954;1:217–22.

[4] Daduna J, Paixão J. Vehicle scheduling for public mass transit, an overview. In: Daduna J, Branco I, Paixão J, editors. Computer-aided transit scheduling. Berlin: Springer, 1995.

[5] Odoni AR, Rousseau JM, Wilson NHM. Models in urban and air transportation. In: Pollock SM, Rothkopf MH, Barnett A, editors. Handbooks in operations research and management science, vol. 6: operations research and the public sector. Amsterdam: North-Holland, 1994. p. 107–50.

[6] Dell'Amico M. Una Nuova Procedura di Assegnamento per il Vehicle Scheduling Problem. Ricerca Operativa 1989;51:3–21 (in Italian).

[7] Bertossi AA, Carraresi P, Gallo G. On Some matching problems arising in vehicle scheduling models. Network 1987;17:271–81.

[8] Carpaneto G, Martello S, Toth P. Algorithms and codes for the assignment problem. Annals of Operations Research 1988;13:193–223.

[9] Paixão J, Branco IM. Bus scheduling with a fixed number of vehicles. In: Daduna J, Wren A, editors. Computer-aided transit scheduling. Berlin: Springer, 1988.

[10] Ahuja RK, Magnanti TL, Orlin JB. Network flows. Englewood Cliffs, NJ: Prentice-Hall Inc., 1993.

[11] Daduna J, Wren A, editors. Computer-aided transit scheduling. Berlin: Springer, 1988.

[12] Liebowitz J, Lightfoot P. Scheduling approaches in expert systems: a study of their applicability. Expert Systems with Applications 1993;34:231–5.

[13] Grant TJ. Lessons for O.R. from A.I.: a scheduling case study. Journal of Operational Research Society 1986;37:41–57.

[14] Kusiak A, Chen M. Expert systems for planning and scheduling manufacturing systems. European Journal of Operational Research 1988;34:113–30.

[15] Kerr RM. Expert system in production scheduling: lessons from a failed implementation. Journal of Systems Software 1992;19:123–30.

[16] Daduna J, Mojsilovich M. Computer aided vehicle and duty scheduling using the hot programming system. In: Daduna J, Wren A, editors. Computer-aided transit scheduling. Berlin: Springer, 1988. p. 133–46.

[17] Ceder A, Fjornes BF, Stern HI. OPTIBUS: a scheduling pachage. In: Daduna J, Wren A, editors. Computer-aided transit scheduling. Berlin: Springer, 1988.

[18] Yu PL. Multiple criteria decision making: five basic concepts. In: Nemhauser GL, Rinnoy Kan AGH, Todd MJ, editors. Handbooks in optimization research and management science, vol. 1: optimization. Amsterdam: North-Holland, 1991. p. 633–99.

[19] CPLEX Optimization, Inc. Using the CPLEX Callable Library – Version 4.0. CPLEX Optimization, Inc., Incline Village NV, 1995.

[20] Bal HE, Grune D. Programming languages. In: Coffman EG Jr, Lenstra JK, Rinnooy Kan AGH, editors. Handbooks in operations research and management science, vol. 3: Computing. Amsterdam: North-Holland, 1992. p. 31–89.

[21] Kowalsky R. Logic for problem solving. Amsterdam: Elsevier Science, 1979.

[22] Krzystof RA. Logic programming. Amsterdam: Elsevier, 1990.

[23] Biethahn J, Nissen V, editors. Evolutionary algorithms in management applications. Berlin: Springer, 1995.

[24] Goldberg DE. Genetic algorithms in search, optimisation and machine learning. Reading, MA: Addison-Wesley, 1989.

[25] Holland JH. Adaptation in natural and artificial systems. Ann Arbor: MIT Press, 1992.

[26] Baita F, Mason F, Poloni C, Ukovich W. Genetic Algorithm with redundancies for the Vehicle Scheduling Problem. In: Evolutionary algorithms in management applications. Berlin: Springer, 1995. p. 341–53.

[27] Horn J, Nafpliotis N. Multiobjective optimisation using the Niched Pareto genetic algorithm. IlliGAL Report no. 93005. University of Illinois at Urbana-Champaign, Illinois Genetic Algorithm Laboratory, Urbana, 1993.

Flavio Baita is a Ph.D. Student of Information Engineering at the University of Trieste, Italy, Faculty of Engineering. His research fields include combinatorial and heuristic optimization, genetic algorithms, routing, scheduling, and logistic system design. He has been a visiting scholar at the University of California at Berkeley.

Raffaele Pesenti is an Associate Professor of Operation Research at University of Palermo (I). He received his Ph.D. in industrial engineering from University of Genova (I). His interests are in the areas of Logistics, Transportation and Measurement of Productive Performances.

Walter Ukovich is an Associate Professor of Operations Research at University of Trieste, Italy, Faculty of Engineering. His major research interests are in optimization theory, logistics, production planning and control, and evaluation. He has published papers in various journals including *Operations Research, SIAM Journal on Algebraic and Discrete Methods*, *SIAM Journal on Optimization*, *IEEE Transactions on Robotics and Automation*, *Networks*, *Journal of Optimization Theory and Applications*, *Transportation Research*, *Naval Research Logistics*, *International Journal of Production Economics*, *European Journal of Operational Research*.

Daniela Favaretto is an Assistant Professor of Operations Research at University "Ca' Foscari" of Venice, Italy, Faculty of Economics. Her major research interests are in optimization theory, Mathematical Programming, Optimal Control. She has published papers in *Dynamics and Control*, *TOP Optimization*.