Metaheuristics

for

The Bus-Driver Scheduling Problem

Helena Ramalhinho Lourenço[*]
Universitat Pompeu Fabra, Spain

José Pinto Paixão
Universidade de Lisboa, Portugal

Rita Portugal
Universidade de Lisboa, Portugal

Abstract

We present new metaheuristics for solving real crew scheduling problems in a public transportation bus company. Since the crews of these companies are drivers, we will designate the problem by the bus-driver scheduling problem. Crew scheduling problems are well known and several mathematical programming based techniques have been proposed to solve them, in particular using the set-covering formulation. However, in practice, there exists the need for improvement in terms of computational efficiency and capacity of solving large-scale instances. Moreover, the real bus-driver scheduling problems that we consider can present variant aspects of the set covering, as for example a different objective function, implying that alternative solution methods have to be developed. We propose metaheuristics based on the following approaches: GRASP (greedy randomized adaptive search procedure), tabu search and genetic algorithms. These metaheuristics also present some innovation features based on the structure of the crew scheduling problem, that guide the search efficiently and able them to find good solutions. Some of these new features can also be applied in the development of heuristics to other combinatorial optimization problems. A summary of computational results with real-data problems is presented.

Keywords: metaheuristics, bus-driver and crew-scheduling, tabu-search, GRASP, genetic algorithms.

[*] Department of Economics and Management, Universitat Pompeu Fabra, R.Trias Fargas 25-27, 08005 Barcelona, Spain (ramalhin@upf.es)

1. Introduction

Public transportation companies are faced with important challenges in the area of transportation planning due mainly to population growth, environmental policies, requirements for a service with quality, and the pressure from governments to a better use of their resources. Therefore, transportation planning systems in public transport have been gaining importance since a large amount of money can be saved if the available resources are employed efficiently, or wasted if not. The same challenges are faced by the private transportation companies. As a consequence, there is an increasing need for computerized tools to aid planners in public and private companies.

Several projects have been developed, or are under developing, to design a Transportation Planning System, as for example HASTUS, Rousseau et al.[1985], IMPACTS, Smith & Wren [1988], HOT, Daduna & Mojsilovic [1988] and TRACS II, Kwan, Kwan, Parker and Wren [1997]. All these systems are used by transportation companies in several countries. This work is also a part of a large project in transportation planning, designated by GIST, developed in INEGI and ICAT, in coordination with five bus transportation companies: CARRIS, STCP, Horarios do Funchal, VIMECA, Barraqueiro and Rodoviaria. GIST is a Decision Support System to assist the planning department of public and private transportation companies or transit authorities in the operations management. The system includes the production of timetables, the scheduling of vehicles, the generation of daily duties for drivers, and the construction of rosters of individual drivers for a certain period.

The transportation planning is decompose in several subproblems due to its complexity: timetabling, vehicle scheduling, crew scheduling and roster scheduling, with relations between them as it can be seen in Figure 1, Freling [1997].

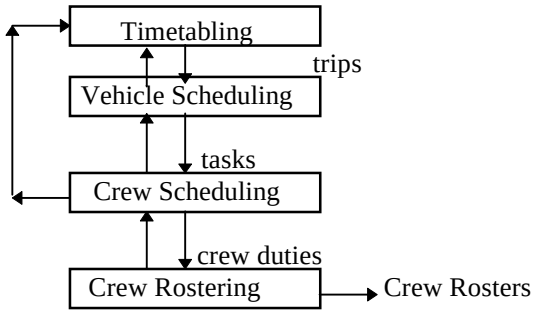

Figure 1. The transportation planning

The bus network structure includes all the information on the company operational network, this information is well known and does not change for large periods of time. The transport service is composed by a set of lines, usually identified by a number, that correspond to a bus traveling between two points in town or between two towns. For each line the respective frequency is determined based on demand. Afterwards, a timetable is constructed, resulting in trips that correspond to a start and to an end point, and a start and end time. The vehicles scheduling assigns vehicles to trips. The crew scheduling problem generates daily duties for drivers. And, the roster scheduling constructs monthly rosters comprising the daily duties of individual drivers together with their days off and holidays. For a survey in transportation planning see Wren and Rousseau [1995], Wren [1996], Odoni et al.[1994], Daduna et al.[1995], and for a recent survey in vehicle and crew scheduling see Freling [1997].

Several Linear Programming methods have been proposed to solve the crew scheduling problem or special formulations of it. These methods have been widely applied by the previous mentioned systems. Actually, the DSS GIST uses an algorithm based on Linear Programming,

Beasley [1987], Portugal [1998], and the Vasko and Wolf [1988] and Beasley [1987] heuristics to obtain lower and upper bounds for the bus-driver scheduling problem.

On everyday planning, companies need fast methods to obtain several good scenarios in real time that can help the decision maker. Therefore, the objective of the paper is to develop methods to solve real crew scheduling problems that can be used in a transportation planning system, in a user-friendly environment. These methods must be flexible in presence of variant aspects of the problem, as for example different objective functions. The methodology followed to develop such a solution approach was the metaheuristics, since they can obtain good solutions in a short time and allow any type of objective function. Therefore, we put emphasis in solving these problems in a reasonable time, and we will take into account the environment of the user when developing the solution methods.

Our main applications are for bus transportation companies, where all our data came from and, since the crew of the buses are drivers, we will designate the problem as the Bus-Driver Scheduling Problem (**BDSP**). However, the metaheuristics presented here can also be applied to other crew-scheduling problems in different sectors, as for example, train and airlines companies.

In the next chapter, we present more details of the bus-driver scheduling problem, and we formulate it as a set covering problem. In chapters 3, 4 and 5 we propose a GRASP, Tabu Search and Genetic Algorithm heuristic, respectively, to solve the BDSP. In chapter 6, a summary of the computational experiment is presented followed by the conclusions and description of future work.

2. The bus-driver scheduling problem

The bus-driver scheduling problem can be stated as finding the minimum cost set of feasible daily duties that cover all trips or vehicle blocks. A vehicle block is the itinerary of a vehicle between its departure from the garage and its returns to the garage. Any vehicle block can be split into pieces of work, such that a split occurs only at a relief point, i.e. a time and a place at which change of drivers is possible. A driver's duty is a set of pieces of work that can be assigned to a driver.

Several formulations have been proposed for the crew-scheduling problem. We will consider an approach based on the set covering formulation of the problem. One of the advantages of this formulation is that it is independent of labor contract and specific company rules. Therefore the generation of all feasible driver duty modules is separated from the selection of the minimal cost or best quality driver duties. In this case, the set of all feasible driver duties is generated by complete enumeration, Agra [1993], i.e. all combinations of pieces of work that complies with labor contracts and company rules are defined previously to the solving of the BDSP. This approach allows adjusting only the first module for each transportation company.

For each duty, a cost is associated which represents real cost, as extra hours, night hours and meal costs, and artificial costs, as for example vehicle change, type of service, number of hours over the average. Other components of the cost can be considered if they are requested by the transportation company.

Let $N = \{1,2,\ldots,n\}$ and $M = \{1,2,\ldots,m\}$ be the index sets for the feasible duties and pieces of work respectively. Let c_j be the cost associated with duty j, and define the matrix A as follows:

$$a_{ij} = \begin{cases} 1 & \text{if the duty index } j \text{ includes the } i\text{-th piece} \\ 0 & \text{otherwise.} \end{cases}$$

Consider the following variables:

$$x_j = \begin{cases} 1 & \text{if } j\text{-th duty is in the solution;} \\ 0 & \text{otherwise.} \end{cases}$$

The bus-driver scheduling can be formulated as a set-covering problem (SCP):

$$\text{Min} \quad z(x) = \sum_{j=1}^{n} c_j x_j \qquad (1)$$

$$\text{s.t.} \quad \sum_{j=1}^{n} a_{ij} x_j \geq 1, \quad i \in M, \qquad (2)$$

$$x_j \in \{0,1\} \quad j \in N. \qquad (3)$$

Constraint (2) means that all piece of work has to be covered by a duty, and can be covered more than once. Columns correspond to duties and the lines correspond to the pieces of work. We say that a line i is covered if exist a column j in the solution such that $a_{ij}=1$. This means that exists a driver's duty j in the solution that contains the piece of work i.

The SCP is NP-hard, Karp [1972], Garey and Johnson [1979]. Several approaches have been proposed to solve the SCP, based on heuristics, column generation, lagrangian and linear programming relaxations and state space relaxation. Recent surveys can be found in Odoni et al.[1994], Freling [1997] and Daduna et al.[1995]. Several greedy heuristics, based on different priority or greedy functions, are presented in Vasko & Wolf [1988]. Beasley and Chu

[1996] and Al-Sultan et al.[1996] have proposed some approaches based on genetic algorithms (GA). Recently, several authors are also presenting GA for the BDSP. Among these are Clement and Wren [1993]. Wren and Wren [1995], Kwan and Wren [1997], Kwan, Kwan and Wren [1997] and Galvão, Sousa and Cunha [1998].

In the previous formulation, the objective is to minimize the total cost of the driver's duties. But, the transportation companies may have different objectives when planning for example the service quality which can be measured in different ways in each company and, therefore, we are looking for a robust and flexible method that can be applied to any objective function. Some examples of measures of the quality of the service are:

- The number of pieces of work not covered. Some companies allow pieces of work or trips to be uncovered, and their main objective is to minimize the number of the uncovered pieces.
- The unfitness value, which measures the amount of infeasibility with respect to the set partition formulation, Chu and Beasley [1995]. We define the unfitness by $u_p = \sum_{i=1}^{m} |w_i - 1|$, where w_i is the number of columns in the current solution x that cover line i. In practice, this value is quite important for the users, since some degree of overcovered is desired, however if a solution has some pieces of work with too many drivers assigned, the planner will have to adjust manually this solution until have one with smaller unfitness value.
- The total number of duties. Some companies claim that a small number of duties respecting the labor and company constraints allow a better planning and easier implementation.

- The total number of duties with only one piece of work (single piece-of-work duty). If we allow duties with only one piece of work, these duties will be very expensive in terms of labor cost. Therefore some companies use them only in special cases and want to minimize the use of this type of duties.
- The number of vehicle changes. The change of a vehicle driver can disrupt the operational functioning of the company, and cause complains from the drivers. Therefore, some companies are mostly worried about minimizing the number of changes.
- Combination of any of the above.

In general we will talk about evaluation functions for the solutions that can be the cost function, or any of the above functions. However, all users in the different companies want to be able to analyze several solutions in real time and change them, if they find the need to do it. This last point is very important for the companies to accept the transportation planning system. For all these reasons, some of the classical methods are not adequate. The recent advances and success in metaheuristics applied to other problems advocates to the use of these techniques as a good way to proceed.

3. GRASP for the BDSP

The Greedy Randomized Adaptive Search Procedure, GRASP, was proposed by Feo & Resende [1989], and since then it has been applied to several Combinatorial Optimization problems with success.

The basic idea of GRASP is to combine constructive methods with local search. In the first phase, the construction phase, a feasible solution is build in a greedy fashion, introducing an

element at each time considering some random choices. At each step of the greedy heuristic, a restricted list of the best elements to be included in the solution is created, then one of these elements is randomly selected and inserted into the solution. The process is repeated until a feasible solution is found. In the second and last phase, a local search method is applied to try to improve the solution found in the first phase. Both phases are repeated until a certain stopping-criteria is verified.

GRASP can be seen has a multi-start sampling algorithm, but instead of starting from purely random solutions, it constructs the initial solution using a greedy adaptive probabilistic heuristic.

The main steps of the GRASP are:

1 While a certain stop-criteria is not verified

  1.1 Get a greedy randomized initial solution *x*

  1.2 Apply local search starting with solution *x*.

2 Return the best solution found.

The first phase of GRASP corresponds to a greedy adaptive randomized heuristic where at each iteration a column is added to the solution; we keep adding columns until all lines have been covered, and it can be described as follows:

1 While there is an uncovered line, repeat

  1.1 Use a greedy function to evaluate the benefit of including each column.

  1.2 Select the three best columns (restricted candidate list).

  1.3 At random, select one column from the candidate list.

  1.4 Include this column in the solution and update the greedy function.

The greedy function used is the following one: c_j / k_j where c_j is the cost of column *j* and k_j is the number of uncovered lines that column *j* covers if added to the solution. The greedy heuristic is adaptive since, at the end of each iteration (step 1.4) the value of k_j is adapted taking into account the columns already included in the solution.

The second phase of the GRASP is a local search heuristic. The most important concept in a local search is the definition of the neighborhood for the problem in consideration. We propose a exchange neighborhood, i.e. remove a column of the solution and add a new column that covers at least one line uncovered. It is importance to notice that, for this neighborhood, the number of columns of the neighbor solution is always equal to the number of columns of the initial solution obtained in step 1.1 of GRASP.

More precisely, let *x* be a solution of the BDSP, and let N_x* the set of column in the solution *x*. Then, the neighborhood of the current solution *x* can be defined as follows:

$$N(x)=\{y: \exists\, l,k \in N, \text{ such that } l \in N_x^* \ (x_l = 1) \text{ and } l \notin N_y^* \ (y_l = 0) \text{ and } k \notin N_x^* \ (x_k = 0)$$
$$\text{and } k \in N_y^* \ (y_k = 1) \text{ , also } \exists\, i \text{ (line)} : \sum_j a_{ij} x_j \le 1 \text{ and } a_{il} x_l = 1, \text{ and } \sum_j a_{ij} y_j \ge 1\}$$

The exchange neighborhood, as described, leads to unfeasible solutions, i.e. some lines are not covered since we only oblige that the entering column covers one uncovered line, but the leaving column can leave one or more lines uncovered. Allowing these extra (unfeasible) solutions is known as strategic oscillation, Glover and Laguna [1997], and provides escapes from local optimal solutions and drives the search to good solutions. To control the trajectories and avoid visiting to many unfeasible solutions, a fitness function that penalizes the

unfeasibility is defined as follows: $f(x) = \sum_{j=1}^{n} c_j x_j - K \times \sum_{i=1}^{m} \min(0, w_i - 1)$ where w_i is the number of columns in the current solution x that cover line i, and $K>0$ is a parameter representing the "cost" of an uncovered line.

Another aspect of the local search is related with the candidate list of neighbors. The feasible solutions for most of our real problems present a very small ratio between the number of columns in the solution and the total number of columns. Therefore, the number of candidate columns to enter is quite large. To avoid considering all possible columns, we start by the ones with smaller penalized cost, defined next. Consider a current solution x, after removing the column l, the penalized cost is obtained by taking in account the uncovered lines and overcovered lines that the column covers after entering the solution. The penalized cost of column k not in the solution is defined as follows: $p_k = c_k + \mathrm{U} \times \sum_{i=1}^{m} u_{ik} + Q \times \sum_{i=1}^{m} q_{ik}$, where $u_{ik} = a_{ik} * \min(0, w_i - a_{il} - 1)$, $q_{ik} = a_{ik} * \max(0, w_i - a_{il})$, $U>0$ and $Q>0$ are parameters associated with lines uncovered and overcovered, respectively, and w_i defined as above. If a column covers many uncovered lines the cost is reduced. On the other hand, if a column covers many lines already covered, the cost increases.

The local search algorithm is a first improvement and can be described as follows:

1 Consider an initial solution

2 While there is an untested neighbor of x:

  2.1 Let x' be the next untested neighbor of x in the candidate list defined previously;

  2.2 If $f(x') < f(x)$ let $x=x'$

3 Return x

The reasons for using such a simple local search and neighborhood are due to the aim of repeating several times the GRASP iteration. Each time starting with a different initial solution, and the application of the GRASP as a subroutine of the tabu search and genetic algorithm, as we will see.

4. Tabu search algorithm for the BDSP

Tabu Search is an adaptive procedure originally proposed by Glover [1986]. Recently, this metaheuristic has been gaining importance as a very good search strategy method to solve combinatorial optimization methods. For a recent survey see Glover & Laguna [1997].

The basic idea of tabu search is to escape from a local optimum by means of memory structures. Each neighbor solution is characterized by the move and short term memory is used to memorize the attributes of the most recent applied moves, incorporated via one or more tabu list. Therefore, some moves are classified as tabu and consecutively some neighbor solutions are not considered. To avoid not visiting a good solution, an aspiration criteria can be considered. At each iteration, we choose the best neighbor of the current solution that it is not tabu or verifies an aspiration criteria. The algorithm stops when a certain stopping-criteria is verified. The best solution found during the search is then the output of the method.

Initially we will present the basic features of a simple tabu search algorithm for the bus driver scheduling problem, and afterwards we will describe some new aspects of the heuristic.

The main steps of the TS are:

1 Obtain an initial solution *x*.

2 While a certain stop-criteria is not verified

  2.1 Obtain a neighbor of *x, x'*, not tabu or satisfying an aspiration criteria with minimal cost among the neighbors of *x*

  2.2 Set *x=x'* and update the tabu list and aspiration criteria.

3 Return the best solution found.

The initial solution can be obtained by two methods: a random initial heuristic and a greedy heuristic. The random initial works as follows: For each line, randomly select a column between the ones that cover it. When all the lines have been considered, the redundant columns are removed. The greedy heuristic builds a solution in a greedy fashion, at each step, a column is selected to enter the solution following some greedy function; repeat this step until all lines have been covered. This is the deterministic version of the greedy adaptive probabilistic heuristic of the previous chapter.

We considered three neighborhoods, the exchange neighborhood, presented before, the remove and the insert neighborhood. The insert neighborhood considers all solutions that can be obtain from the current solution by the introduction of one column. In the opposite way, the remove neighborhood considers all solutions that are obtained from the current one by removing one column. Most of the solutions consist of a small number of columns. Therefore, the insert neighborhood is larger than the remove neighborhood, since there is a large number of candidate columns to enter the solution that to be removed. Due to the size of the insert neighborhood, we consider a candidate list strategy. The strategy makes the search more aggressive and avoids visiting not so good solutions. The candidate list is build considering the

following rule: a column is considered for entering if covers one uncovered or single covered line and if its penalized cost, p_k, is less or equal than the average cost.

For each of the three neighborhoods a certain number of iterations are performed, which depend on a parameter related to the size of the neighborhood. The order of the neighborhood search is as follows: insert neighborhood, exchange neighborhood, remove neighborhood, exchange neighborhood, and repeat.

To each solution we calculate its value using the fitness function, *f(x)*, described before, since we can have unfeasible solutions. Other evaluation functions related to the quality of the service or combinations could be used.

Two tabu list were considered: the insert tabu list and the remove tabu list. The remove tabu list contains all columns that have been inserted recently in the solution and therefore they cannot been removed. The insert tabu list contains the columns that have been removed in the most recent iterations, and therefore it is tabu to insert them again in the solution. The two tabu list have different size because the candidates columns for each one is quite different. The size of the insert tabu list is a percentage of number of columns in the first solution, and smaller than the size of the remove one, which is a percentage of the total number of columns.

Finally, the aspiration criteria used was the most common one: the tabu status is overruled if the neighbor solution has a objective function value smaller than the best found up to that iteration

Next we present some innovation aspects considered in the tabu search. The inclusion of these new features has the objective of improving the search, and can be seen as intensification strategies based on classical optimization and hybrid methods.

Suppose we apply the tabu search for several iterations using only the insert neighborhood. The resulting solution will have a large number of columns, and each line will be covered by several columns. To obtain good solutions with fewer columns and, such that each line is not overcovered, we can apply an exact method to set covering subproblem using the cost function. Note that the subproblem has a smaller dimension compared with the full problem, since we are only considering the columns in the current solution. For small instances we can apply an enumeration method. For larger instances we apply the GRASP method described previously since the exact method takes too much time. This permits us to find good solutions, eliminating the most expensive columns, which are inserted in the insert tabu list. Moreover, this intensification strategy allows us to obtain solutions that would be difficult to find by the usual search, and the computational time did not increase significantly.

5. Genetic algorithm for the BDSP

Genetic Algorithms (GA) were originally developed by Holland [1975] and are intelligent search heuristics based on evolution. The basic idea of the genetic algorithms is that, during the course of evolution, the best fitted individuals have better change to survive and reproduce, meanwhile the least fitted individuals will be eliminated. A GA simulates this behavior by taking into account a initial population of solutions (individuals) and fitness function, usually associated with the objective function, and by means of some selection

techniques and operators, this population is replaced by a new one, with higher fitness. This cycle is repeated until a satisfactory solution is found.

The fundamental aspects of genetic algorithms are: the representation of the solutions, parent selection, population replacement and the genetic operators, crossover and mutation.

GA has been applied to a wide range of problems in several areas. For a survey in GA see Davis [1996]. The GA proposed to solve the BDSP is based on the work of Beasley and Chu [1996], but several aspects have been adapted to the specific problems we want to solve and their environment.

The main steps of the GA are:

1 Generate a family of trial solutions.

2 Calculate the fitness of each solution.

3 While a certain stop-criteria is not verified

  3.1 Select elements from the population

  3.2 Crossover these elements.

  3.3 Mutation some elements

  3.4 Population replacement

4 Return the best solution found.

Solutions are represented as a binary vector of dimension *n*, indicating if the column (driver duty) is or is not in the solution. This is the obvious representation of a covering solution, but others have been suggested, see Beasley and Chu [1996]. The major problem of this representation is that the application of a crossover or a mutation operator frequently

produces an unfeasible solution. To control the unfeasibility, we use the fitness function described previously. Also, if necessary, a greedy heuristic can be applied to restore feasibility.

The initial population is generated by different methods to guarantee some diversification. The first method generates most of the population, except for 10 solutions, and can be described as follows: for each line, choose randomly a column that covers it, and them apply a simple heuristic to eliminate columns that only cover already covered lines. The remain 10 solution are obtained by the heuristics described in Vasko and Wolf [1985], so we guarantee the presence of some good solutions in the initial population and making the search to converge faster. The initial population has 100 solution and, as the offsprings are introduce the population, it can grow until 200. When this limit is obtained, we choose the 100 best solutions and eliminate the remain ones

The parent are selected by a tournament selection based on the uniform probability function and on the evaluations functions. In this method, two groups of T solutions are selected uniformly from the population and the best solution of each group is selected for parent. Beasley and Chu [1996] refer that this method is one of the more efficient methods and suggest the value 2 for T. To determine the best solution of each group, it can be used different evaluation functions, as the cost function, the fitness function or any quality service measure as described in section 2. To diversify the search, at some stages, we randomly choose one of the evaluation functions. The reason for having this approach is because the main objective is to obtain different scenarios for the BDSP, not only to obtain the minimal cost solution.

The crossover operator defines the way of combining two or more parent solutions in a offspring solution. Several method have been considered as the one-point and two-point crossover, the uniform crossover based on the uniform distribution and the generalized fitness-based crossover or fusion operator, see Beasley and Chu [1996]. We consider the two-point crossover, where two crossover points are selected randomly and the segments of the parents are swapped to produce two offspring solutions.

An improvement in the GA was obtained by defining a new crossover operator that we denominate by perfect offspring. These operator considers two parents and try to obtain the best offspring of these parents by solving a set covering subproblem, where all the lines are considered but only the columns present in the parent solutions are taking in account. We follow the intensification strategy described in previous section, applying exact methods for very small instances and the GRASP method for larger ones. This approach follows the methodology known as memetic algorithms, Moscato [1989] and a similar approach based on Integer Linear Programming was used by Clement and Wren [1993]. The memetic algorithms are a population-based global search that also considers local search heuristics applied to each individual, and, in this way, combine several aspects of different methods to improve the performance of the search. Our GA, not only combines aspects of different search methods, but also optimization methods based on the structure of the problem. The perfect offspring crossover allows the search to converge rapidly to good solutions and this idea can be applied to other combinatorial optimization problems.

The mutation operator permits to introduce random variations in the solutions, and play an important role in the capacity of the GA to diversify the search, especially when all the solutions in the population become similar. A mutation is applied to each offspring solution

after crossover. We use the mutation operator proposed by Beasley and Chu [1996] based on removing or including a column, following the mutation rate presented next:

$$\text{numinv} = \left\lceil \frac{m_f}{1+\exp(-4m_g(t-m_c)/m_f)} \right\rceil$$

where: t = number of offspring solutions that have been generated; m_f = final stable mutation rate; m_c = number of offspring solutions generated at which a mutation rate $m_f/2$ was reached; m_g = gradient at $t=m_c$;

After the crossover and mutation, a offspring solution can be or not included in the population, following the next heuristic. Iteratively, compare the child solution with the solutions in the population starting at a random position. Three situations can occur:

1. A solution having better values for all evaluation function is found. Discard this child solution, and go back an obtain a new one.
2. A solution having worst values for all evaluation function is found. The child solution substitutes this solution.
3. If none of the above happens, then the child solution is not dominated by any solution in the population, so add the child solution to the population, without removing any solution.

The last added offspring can be a parent in the next iteration.

For the GRASP and Tabu Search methods, we used the fitness function to evaluate the solutions, however these methods allow to take in consideration other evaluation function if proposed by the transportation company. For the Genetic Algorithm, we went a little further and, at some stages of the search, we apply a diversification strategy by considering several evaluations functions based on the service quality, and not only in the cost or fitness function.

At the end of the search, the best solutions found for each of the active evaluators are output and the user has access to all of them.

6. Summary of the computational results

The motivation for this research was to develop solution methods to solve real crew scheduling problems that can be used in a transportation planning system, in real-time and in a user-friendly environment. Therefore, the computational experiment was design to analyze the performance of the different metaheuristics previously described when applied to real instances. Our main objectives are to gain understanding on the behavior of the different methods and compare the performance of these ones with method actually used by GIST. We present a summary of the extensive computational testing reported in Portugal [1998].

All numerical tests were carried on a Workstation IBM Risc 600 with 128 MB Ram memory. The algorithms were coded in C. Five real data-sets obtained directly from the companies associated with the project GIST are considered, see Table 1. The density of matrix A is between 2% and 6%.

Table 1: Size of the test problems.

	Group 1				
	S1	**RL1**	**F2**	**RL2**	**RL3**
m	98	215	141	348	348
n	2002	8511	30842	25790	74019

In Table 2 we present the results obtained by the LP method actually used by the GIST transportation system. The results for the RL3 where obtained by other methods, due the actual GIST system is not able to solve such a large instance. The remaining results are presented in the following tables: the GRASP method are presented in Table 3, the Tabu

Search heuristic in Table 4, and two versions of the Genetic Algorithms in Tables 5 (simple GA, two point crossover) and 6 (perfect offspring crossover). For each test problem we present the value of the cost function, the following evaluation functions: unfitness, total number of duties, the number of single piece-of-work duties, and the time to find the best solution.

Table 2: Computational results of the LP-Based Method.

	Opt./UB	LB	Unfitness	N. of Services	N. of single p.o.w. duties	Time seconds
S1	282705*		25	48	9	5
RL1	467254	426888	68	63	6	326
F2	167321	159250	117	28	0	1150
RL2	470017	381170	236	66	1	2220
RL3*	453103	370307	-	64	-	2816

Table 3: Computational results of the GRASP Heuristic.

	Cost	Unfitness	N. of Services	N. of single p.o.w. duties	Best Time
S1	332123	39	59	0	95
RL1	482876	51	66	6	908
F2	165244	60	28	1	2614
RL2	480632	146	69	1	2173
RL3	459644	108	67	1	3608

Table 4: Computational results of the Tabu Search Heuristic.

	Cost	Unfitness	N. of Services	N. of single p.o.w. duties	Best Time
S1	209816	21	51	0	11
RL1	466797	39	53	0	593
F2	166028	41	22	0	351
RL2	507306	97	64	0	186
RL3	479870	75	68	0	3635

The tabu search outperformed the remaining methods, since overall it obtained the best results for all the evaluations functions, however a few final solutions where unfeasible and it was necessary to apply a heuristic to restore feasibility. Overall methods, the GRASP obtains the worst results.

Comparing the LP-based solutions with the TS-solutions, we can observe that these last solutions have better values with respect to the three evaluations functions, meanwhile the cost is a little higher. In average, the running times are similar for both methods. Showing the both types of solutions to the final users, they comment that the TS-solutions have the characteristics they are looking for: less number of single piece-of-work duties, smaller unfitness, less or approximately the same number of services and similar cost.

Table 5: Computational results of the Genetic Algorithm (GA6).

	UB	Unfitness	N. of Services	N. of single p.o.w. duties	Best Time
S1	735092	24	53	2	13
RL1	463840	38	61	6	165
F2	177644	82	29	0	1144
RL2	478586	139	68	1	338
RL3	452482	102	64	1	540

Table 6: Computational results of the Genetic Algorithm-2 (GA9).

	Opt./UB	Unfitness	N. of Services	N. of single p.o.w. duties	Best Time
S1	711781	17	52	0	202
RL1	473084	42	62	6	3608
F2	177644	83	29	0	2397
RL2	481780	132	66	1	3600
RL3	456825	101	66	1	3623

Considering the GA and the LP-based solutions, we can observed that, if the GA runs for a short time, we can obtain similar solutions to the LP ones. The GA-solutions have small values for the number of single piece-of-work duties and smaller unfitness value, approximately the same number of services, but higher cost. For larger running times, the results improve for all evaluation functions.

Between the two versions of the genetic algorithms, we can observe that the perfect offspring crossover version (GA9) obtains slightly better results but at expensives of high computational running times. Therefore, some more work is needed to obtain a faster perfect offspring crossover.

Users of different companies were asked to evaluate the performance of the different methods. They point out the importance of the evaluations functions, other than the cost function, and the advantage of being able to obtain several scenarios as the main benefit and utility of the new metaheuristics. As a consequence of having good values for the unfitness and single piece-of-work duties, the need for manual adjustments that yield satisfactory solutions is reduced significantly leading to a better acceptance of the GIST system.

7. Conclusions and future work

In this paper we have presented several metaheuristics for solving the bus-driver scheduling problem in transportation companies. These methods have been incorporated in the Decision Support System for Transportation Planning GIST allowing to solve very large-scale problems in real time, substituting the previous LP-based method that was object of several criticism by the users.

Our computational testing showed that the Tabu Search and the Genetic Algorithms leads to good results within reasonable times, and the results compare favorably with the actual LP-based solutions. Creating real schedules, meeting the requirements of the final users shows the success of this approach. Moreover, the system GIST using these new methods to build bus

driver schedules permits the achievement of a final solution that do not need manual adjustment by the user, a common practice for the LP-based solutions.

The system GIST can be used for operational decisions, helping the every day planning or as strategic asset. During the negotiations of union contracts or changes in transportation regulation, users can successfully manage the resources gaining a sustainable competitive advantage. A similar approach is described in Campbell et al.[1997] for the airline industry.

As future work, we would like to consider more structure elements of the problems in the metaheuristics. Since the set covering problem has been very well studied, the use of this knowledge in the design of a metaheuristic can improve the search in terms of quality of the solution and running times. Some of the ideas that we are currently working consider the mix of column generation techniques with tabu search or genetic algorithms. This framework is the next step of the approach used to improve the tabu search and the genetic algorithms by considering strategic intensifications based on exact methods and the perfect offspring in genetic algorithms for larger instances. Also, as future research topic, we are interested in developing metaheuristics for the integration vehicle and crew scheduling problems.

Acknowledgments

Thanks are due to PRAXIS (/2/2.1MAT/139/94) for providing funding support for the work described in this paper.

References

1. K.S. Al-Sultan, M.F. Hussain and J.S. Nizami, “A Genetic Algorithm for the Set Covering Problem”, *JORS* **47:** 702-709 (1996).

2. J.E. Beasley and P.C. Chu, “A genetic algorithm for the set covering problem”, *European Journal of Operational Research* **94**: 392-404 (1996).

3. P.C. Chu and J.E. Beasley, "A genetic algorithm for the set partition problem", *preprint* (1995).

4. K.W. Campbell, R.B. Durfee and G.S. Hines, "FedEx Generates Bid Lines using Simulated Annealing", *Interfaces* **27:** 1-16 (1997).

5. R. Clement and A. Wren, Greedy Genetic Algorithms, Optimising mutations and Bus Driver Scheduling, 6th. International Workshop on Computer Aided Scheduling of Public Transportation, Lisboa (1993).

6. J.R. Daduna, I. Branco and J. Paixão, editors, "Computer-Aided Transit Scheduling", *Proceedings of the Sixth International Workshop*, Springer-Verlag, Berlin (1995).

7. J.R. Daduna and M. Mojsilovic, Computer-aided vehicle and duty scheduling using HOT programme system, in J.R. Daduna & A. Wren (eds) *Computer-Aided Transit Scheduling*, Springer-Verlag, Berlin, 133-146 (1988).

8. L. Davis, *Handbook of Genetic Algorithms*, Van Hostrand Reinhold, New York (1996).

9. T.A. Feo and M.G.C. Resende, "Greedy randomized adaptive search heuristic", *Journal of Global Optimization* **6:** 109-133 (1995).

10. R. Freling, "Models and techniques for integrating vehicle and crew scheduling", Ph.D. Thesis, *Erasmus University*, Rotterdam, Nederland (1997).

11. T. Galvão, J. Pinho de Sousa and J. Falcão e Cunha, "Genetic algorithms for the crew scheduling problem: a real experiment with relaxation models", preprint (1998).

12. M.R. Garey and D.S. Johnson, *Computers and Intractability - A Guide to the Theory of NP-Completeness*, Victor Klee Editor (1979).

13. F. Glover, "Future paths for integer programming and links to artificial intelligence", *Computers & Operations Research* **5:** 533-549 (1986).

14. F. Glover and M. Laguna, *Tabu Search*, Kluwer Academic Publishers, Norwell, Massachusetts (1997).

15. J. Holland, *Adaption in Natural and Artificial Systems*, University of Michigan Press, Michigan (1975).

16. R.M. Karp, "Reducibility among combinatorial problems", in *Complexity of Computer Computations*, Plenum Press, New York (1972).

17. A.S.K. Kwan, R.S.K. Kwan, and A. Wren, "Driver scheduling using Genetic Algorithms with embedded combinatorial traits", in *Preprints of the 7th International Workshop on Computer-Aided Scheduling of Public Transportation*, Boston, U.S.A. (1997).

18. A.S.K. Kwan, R.S.K. Kwan, M.E. Parker and A. Wren, "Producing train driver schedules under operating strategies", in *Preprints of the 7th International Workshop on Computer-Aided Scheduling of Public Transportation*, Boston, U.S.A. (1997).

19. R.S.K. Kwan, and A. Wren, "Hybrid genetic algorithms for the bus driver scheduling", preprint (1997).

20. M. Laguna, "A heuristic for production scheduling and inventory control in the presence of sequence-dependent setup times", Graduate School of Business, University of Colorado, Boulder (1997).

21. P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms", Caltech Concurrent Computation Program, C3P Report 826 (1989).

22. A.R. Odoni, J.M. Rousseau and N.H.M. Wilson, "Models in urban and air transportation", in *Operations Research and the Public Sector*, S.M. Pollock, M.H. Rothkopf and A. Barnett (eds), volume 6 of *Handbooks in Operations Research and Management Science*, North-Holland, Amsterdam, 129-150 (1994).

23. A. Paias and J. Paixão, "State space relaxation for set covering problems related to bus driver scheduling", *European Journal of Operational Research* **71:** 303-316 (1993).

24. R. Portugal, "Metaheuristics for the Bus-Driver Scheduling Problems", Msc. Thesis, Faculdade de Ciências da Universidade de Lisboa, Portugal (1998).

25. J.M. Rousseau, R. Lessard and J.Y. Blais, "Enhancements to the HASTUS crew scheduling algorithm", in J.M. Rousseau (ed.) *Computer Scheduling of Public Transport -2*, North-Holand, Amsterdam, 295-310 (1985).

26. J.M. Rousseau, *Computer Scheduling of Public Transport -2*, North-Holand, Amsterdam (1985).

27. B.M. Smith and A. Wren, "A bus driver scheduling system using a set covering formulation", *Transportation Science*, **22A:** 97-108 (1988).

28. F.J. Vasko and F. E. Wolf, "Solving Large Set Covering Problems on a Personal Computer", *Computers & Operations Research* **15:** 115-121 (1988).

29. A. Wren and D.O. Wren, "A genetic algorithm for public transport driver scheduling", *Computers and Operations Research* **22:** 101-110 (1995).

30. A. Wren and J.M. Rousseau, "Bus driver scheduling - an overview ", in: Daduna, J, Branco, I and Pinto Paixao, J (editors) *Computer-Aided Transit Scheduling*, Springer Verlag, 173-187 (1995).

31. A. Wren, "Scheduling, timetabling and rostering - a special relationship?", in: Burke, E K and Ross, P (Editors), *Practice and Theory of Automated Timetabling*, Springer Verlag, 46-75 (1996).