INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

Intl. Trans. in Op. Res. 20 (2013) 443–470
DOI: 10.1111/itor.12013

Enhanced genetic algorithms for a bi-objective bus driver rostering problem: a computational study

Ana Respício[a,c], Margarida Moz[b,c] and Margarida Vaz Pato[b,c]

[a] *Universidade de Lisboa, Departamento de Informática, Bloco C6, 1749-016 Lisboa, Portugal*
[b] *ISEG, Universidade Técnica de Lisboa, R. do Quelhas 6, 1200-781 Lisboa, Portugal*
[c] *Centro de Investigação Operacional, Bloco C6, 1749-016 Lisboa, Portugal*
E-mail: respicio@di.fc.ul.pt [Respício]; mmoz@iseg.utl.pt [Moz]; mpato@iseg.utl.pt [Vaz Pato]

Received 28 August 2012; received in revised form 23 January 2013; accepted 6 February 2013

Abstract

In this work, the bus driver rostering problem is considered in the context of a noncyclic rostering, with two objectives representing either the company or the drivers' interests. A network model and a proof of the NP-hardness of the problem are presented, along with a bi-objective memetic algorithm that combines a specific decoder with a utopian/lexicographic elitism, a strength Pareto fitness evaluation, and a local search procedure. By taking real and benchmark instances the computational behavior of the memetic algorithm is compared with simpler versions to assess the effects of the embedded components. The developed algorithm is a valuable tool for bus companies' planning departments insofar as it yields at low computing times a pool of good quality rosters that reconcile contradictory objectives. This study shows that simple enhancements in standard bi-objective genetic algorithms may improve the results for this difficult combinatorial problem.

Keywords: bus driver rostering; personnel scheduling; evolutionary algorithms; multi-objective heuristics

1. Introduction

Rostering problems arise in several operational contexts, such as transport companies and health care institutions. Rostering deals with the assignment of duties to workers along a planning horizon of a specified length, usually four or more weeks. Ernst et al. (2004) extensively surveyed bibliographical references on personnel scheduling and rostering. More recent studies dealing with rostering in transport companies include Mesquita et al. (2011) for buses, Lezaun et al. (2006) and Sodhi and Norris (2004) for the subway, Dornberger et al. (2008) and Hartog et al. (2009) for railways and Lučić and Teodorović (2007) and Maenhout and Vanhoucke (2010) for airlines.

The rostering of bus drivers has not attracted as much attention from researchers as bus crew scheduling (which produces daily crew duties). Nevertheless, the literature presents some different models for bus driver rostering and other similar personnel rostering problems within the transport domain, namely, network and covering/partitioning formulations. Multilevel network assignment models were proposed in Carraresi and Gallo (1984) and Bianco et al. (1992) for bus driver rostering, and a multigraph model was devised by Caprara et al. (1998) focusing on railway crew rostering. More recently, Cappanera and Gallo (2004) dealt with rostering in an airline company by using a multi-commodity flow network model. Few covering/partitioning formulations are found in the literature (Catanas and Paixão, 1995; Freling et al., 2004; de Matta and Peters, 2009). All these transport domain references proposed single objective optimization methods for the problems or, when dealing with multi-objective problems, they do not explicitly take into account the multi-objective nature of rostering in their solution approaches. In Catanas and Paixão (1995), for example, a bi-objective bus driver rostering model based on a set covering formulation was proposed. The model considers minimizing the maximum roster duration and minimizing the total roster cost. However, the solution approach is focused on a single objective.

It is undeniable that, in any personnel scheduling context, the conflicting interests of both employer and employees must be considered. This requires the adoption of methodologies specifically designed to tackle multiple objectives (Collette and Siarry, 2004). Moreover, the impact of the roster quality in worker performance is recovering the importance that it had in the seventies/eighties of the last century within the personnel rostering literature (Alward and Monk, 1993; Paech et al., 2010). Therefore, solutions should encompass the trade-off between these two axes. Multiple objectives have been considered in various rostering issues, namely bus driver rostering (Emden-Weinert et al., 2001; Moz et al., 2009), airline/train crew rostering (Burke et al., 2010; Caprara et al., 1998; Dornberger et al., 2008; Lučić and Teodorović, 2007), and airport staffing or nurse rostering (Chu, 2007; Landa-Silva and Le, 2008; Moz and Pato, 2007).

However, each case has its own specificities beyond the common core of rostering constraints, which is why the study of multi-objective rostering problems still remains a challenging issue. Evolutionary algorithms have proved to be a powerful, flexible and widely applicable technique to solve both single and multi-objective combinatorial problems. The reference book by Coello, Lamont and Van Veldhuizen covers the main issues of contemporary multi-objective evolutionary algorithms and their applications (Coello et al., 2007).

In this paper, a bi-objective version of the Bus Driver Rostering Problem (BRP) is addressed and a new memetic algorithm (henceforth called MASP) is proposed and evaluated. Section 2 describes BRP in the real context of a bus company. This problem is formulated as a bi-objective multi-commodity network flow model on which the proof of the NP-hardness of BRP is based. Section 3 briefly introduces multi-objective evolutionary algorithms (MOEAs) and section 4 presents the algorithm MASP. Computational results are reported and discussed in section 5, where MASP performance is assessed through standard quality measures for multi-objective approaches and is compared with versions derived by separately removing specific features: local search and utopian/lexicographic elitism. In conclusion, section 6 presents some final remarks.

2. The bus driver rostering problem

2.1. Problem description

The Bus Driver Rostering Problem is here defined in compliance with the institutional requirements and norms of a Portuguese urban bus company, together with the Portuguese Labor Law and the drivers' union contracts. A noncyclic rostering context is considered, as opposed to the usual cyclic rostering in bus companies (Burke et al., 2010; Pedrosa and Constantino, 2001), because we are dealing with a pool of drivers who perform the non-regular tasks and also substitute permanent drivers whenever they are absent. In fact, noncyclic driver rostering increases flexibility which, on the one hand, may match drivers' preferences and, on the other, may be better tailored to cover irregular demand, simultaneously saving salary costs (Montalva et al., 2010).

In the particular situation we have analyzed, the company enrolls a pool of drivers, V, operating daily from 6:00 a.m. to 12:00 p.m. This set of drivers must be assigned to a previously determined set of duties, during a given period—the rostering period, here comprising 28 days or 4 weeks. A duty is a daily working period to be carried out by a single driver on a specific day and it defines a sequence of pieces of work, which may have breaks and idle times in between. Every day, each driver is assigned to work on a particular duty or gets a day off. Hence, for each driver a schedule must be drawn up, that is, a sequence of duties and days off for the rostering period. The BRP thus calls for a roster—a set of schedules for all the bus drivers. The roster must comply with the conditions and rules imposed by labor union contracts, institutional and legal requirements which are viewed as hard constraints. These requirements usually concern the number of days off per week, specific days off per week, a minimum number of Sundays/weekends off in the rostering period, a minimum and a maximum number of days for the length of a rest period, a minimum number of rest hours between two consecutive work periods and a minimum and a maximum number of consecutive working days. Here, the BRP considers the specific hard constraints described below:

(h1) each duty of $T = \cup_{h=1}^{28} T_h$ (with T_h representing the set of duties to be covered on day h) must be assigned to one and only one driver from V;
(h2) each driver from V must be assigned to one and only one duty or a day off, on each day of the rostering period;
(h3) the minimum length of the rest period between consecutive duties that drivers are entitled to is imposed by the definition of sets T_{ih} ($h = 1, \ldots, 28$; $i \in T_{h-1}$) containing the duties that any driver can be assigned to on day h, given that he/she performed duty i on the previous day;
(h4) drivers must not work more than g consecutive days;
(h5) drivers must get at least d_w days off per week;
(h6) drivers must get at least d_s Sundays off in the rostering period;
(h7) the maximum workload per week is b_1 hours;
(h8) the maximum workload per rostering period is b_2 hours.

Note that constraints (h3) and (h4) assume that some assignments of the previous rostering period are known, considering $h = 0$ as the last day of the previous period and counting down for data of previous days up to day (1–g). These data are given by e^v_{ih} as defined in the Appendix. In addition,

constraints (h7) and (h8) take into account that the length of each duty i on day h, represented by t_{ih} hours ($h = 1, \ldots, 28$; $i \in T_h$), is known.

With regard to rostering goals, good rosters for company management generally present low cost assignments. But the company must also consider the interests of its workers. It is well known that the provision of satisfaction to workers enhances the quality of their performance and reduces the level of absences (Ybema et al., 2010). Good rosters for the drivers are usually characterized by equity in the distribution of workload among drivers, Sundays/weekends off, overtime work, and late duties.

In this work, the search for attractive rosters is guided by two objectives that rosters should satisfy as far as possible. *Objective 1* aims at minimizing the value of function f_1 which represents the workload of the driver with the maximum total workload during the rostering period. The total workload of a driver is the sum of the length of the duties assigned to that driver in the 28 days, therefore, $f_1 = \max_{v \in V} \sum_{h=1}^{28} \sum_{i \in T_h \cap \{\text{duties assigned to } v\}} t_{ih}$. This objective implicitly entails a fair distribution of workload among the drivers.

Objective 2 aims at minimizing the value of objective function f_2 which represents salary costs:

$$f_2 = \sum_{h=1}^{28} \sum_{v \in V} \left(\sum_{i \in T_h \cap \{\text{duties assigned to v}\}} c^v t'_{ih} \right) + c^S \times \max\{0, \text{number of drivers assigned to duties} - nd\},$$

where t'_{ih}, the overtime associated with duty i, is equal to $\max\{0, t_{ih} - \bar{t}\}$ ($h = 1, \ldots, 28$; $i \in T_h$) and $\bar{t}$ is the contractual daily working time of a driver. Function f_2 has two components: the first one is the sum of the overtime costs of all drivers of the pool, assuming that the cost of the overtime hour of driver v is known and equal to c^v ($v \in V$); the second component is the sum of the base salaries for extra drivers, assuming that the salary cost per extra driver, c^S, is known. This component is due to the assumption that a specific set of nd drivers from the pool V has already been hired and their base salaries are paid in any circumstance. Hence, the relevant costs to be considered here are the base salaries of extra drivers, that is, the drivers above nd if assigned to duties.

These objectives are of a contradictory nature. In fact, *Objective 2* favors increasing the workload of the cheaper drivers and reducing the number of drivers assigned to work, while *Objective 1* tends to distribute the workload among all the drivers of pool V. Optimization is here considered within the Pareto perspective, that is, we aim at determining the efficient solutions which correspond to the non-dominated points in the objective space.

To sum up, given a set of previously known duties, the aim of BRP is to build a roster for a four-week period, one which satisfies constraints (h1) to (h8) and optimizes the above explained functions:

minimize (f_1, f_2), where

f_1 = workload of the driver with the maximum workload in the rostering period;
f_2 = total cost of overtime per rostering period plus cost of the extra drivers (above nd).

The BRP is an optimization problem that can be formulated within bi-objective binary programming, as presented in the Appendix. This model is similar to the one presented by Moz et al. (2009), although here the problem differs as regards the objectives. In Moz et al. (2009), the

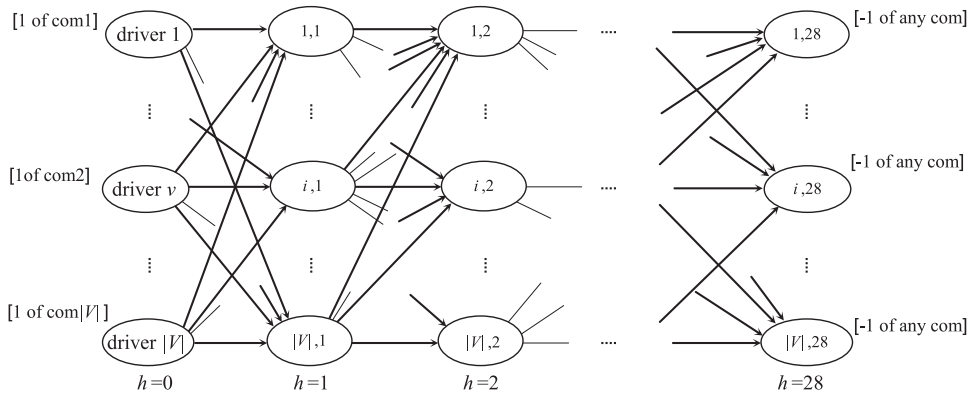

Fig. 1. The rostering network.

first objective was to minimize the maximum total overtime during the rostering period per driver, and the second one was to minimize the number of drivers whose workload is positive but less than q duties, the contractual number of duties per rostering period.

2.2. BRP and NP-hardness

The BRP is an NP-hard problem. To prove this we will model it as a bi-objective multi-commodity network flow problem defined in a multilayer network similar to the one for nurse rostering found in Moz and Pato (2007).

Consider a network with 29 layers of nodes: one per day of the rostering period (the day-layers from 1 to 28) and one additional layer of nodes to initialize the rostering process, layer 0. Each layer has $|V|$ nodes, one node per duty to be covered on the respective day and the remaining nodes for days off on that day. Each arc in this network links pairs of nodes in consecutive layers. Each driver is associated with a commodity and the amount supplied by the respective node at layer 0 is 1 unit of the respective commodity, whereas at the last layer, layer 28, the demand at all the nodes is 1 unit of any of the commodities. Note that the first level of arcs, from layer 0 to layer 1 of nodes, imposes the specific duties each driver can perform on the first day, as a result of the work already performed during the previous week. The other 27 levels include arcs imposing the possible duty sequences. Figure 1 displays the rostering network.

In the above multilayer network, a roster is represented by a $|V|$-commodity binary flow that is disjoint on the nodes and links the nodes of layer 0 to the nodes of layer 28. This flow must satisfy all the constraints (h1) to (h8). Constraints (h1), (h2) and (h3) are naturally imposed by the design of the network. In fact, as each commodity flows from the first to the last layer through disjoint nodes, (h1) are respected. Moreover, the absence of arcs linking nodes of the same layer enforces constraints (h2). Finally, the arcs linking a driver node at layer 0 with duty nodes of day 1 are defined only in case the minimum rest period is respected. The arcs linking duties on consecutive days always correspond to feasible sequences; consequently, constraints (h3) are naturally imposed by the network structure. Constraints (h4) to (h8) are additional global restrictions to be satisfied by the flow.

The BRP consists of determining, through the above defined multilayer network, a multi-commodity binary flow that is disjoint on the nodes, while satisfying additional constraints (h4) to (h8) and minimizing the maximum workload and cost.

Proposition. *The feasibility BRP is NP-complete.*

Proof. Consider the BRP instances where all drivers of the pool V have had a day off on the last day of the previous period, where $g = 28$, $d_w = 0$, $d_s = 0$, $b_1 = 7t^{\max}$ and $b_2 = 28\,t^{\max}$, with $t^{\max} = \max_{h=1,\ldots,28;\, i \in T_h} \{t_{ih}\}$. With these particular settings constraints (h4) to (h8) are redundant. Moreover, as in these instances every node of layer 0 is linked to all nodes of layer 1, due to the last day hypothesis, constraints (h3) are redundant for the first day. Hence, nodes of layer 0 as well as the respective outgoing arcs can be eliminated from the network.

Determination of a feasible solution for this particular BRP corresponds to the determination of $|V|$ paths disjoint on the nodes through the 28 layer network which is a solution of the 28-dimensional matching. In fact, considering the set $M \subseteq X^1 \times X^2 \times \cdots \times X^{28}$ where $X^1, X^2, \ldots, X^{28}$ are disjoint sets having the same number of elements, $|V|$, the 28-dimensional matching is the problem of deciding if M contains a 28-matching, i.e., a subset $M' \subseteq M$ such that $|M'| = |V|$ and no two elements of M' agree in any coordinate. Then the 28-dimensional matching is polynomially transformable into the feasibility BRP.

Since the three-dimensional matching is a well known NP-complete problem (Garey and Johnson, 1979), the 28-dimensional matching is NP-complete, and from the above reasoning the feasibility BRP is also NP-complete. □

From the proposition one infers that the BRP problem is NP-hard.

Besides the computational complexity of the BRP, computational experience has pointed to difficulty in inducing real BRP instances to reach optimality and even to obtain feasible solutions in some cases. In a previous study from Mesquita et al. (2011), using the software CPLEX 11 optimizer (ILOG, 2007) for a BRP single objective model with an objective function mathematically identical to f_1, exact solutions were obtained only for some of the smallest test instances to be described in section 5. This experience, along with the hard theoretical complexity of the problem, prompted us to develop evolutionary heuristics which, in general, are well equipped to deal with high complexity multi-objective problems.

3. Multi-objective evolutionary algorithms

3.1. Context

Early approaches to multi-objective problems consist of combining the objectives under consideration in a single objective function by assigning weights to the objectives and optimizing that function. However, these approaches are unable to reach some of the non-dominated points, the non-supported ones. One important aspect of multi-objective problems is the multiplicity of solutions that must be taken into account. This variety is important for decision-makers whose aim is to analyze other implicit criteria, often enclosed and hard to translate into mathematical formulation. Multi-Objective Evolutionary Algorithms (MOEAs) show the ability to deal with the inherent combinatorial complexity providing, at the same time, points that are potentially non-dominated,

corresponding either to potentially supported or to non-supported solutions. MOEAs also produce a multiplicity of such points, which may provide good approximations to the optimal sets.

In the last decades, several MOEAs have been proposed in the literature. The Strength Pareto Evolutionary Algorithm (SPEA2) (Zitzler et al., 2002) and the Non-dominated Sorting Genetic Algorithm (NSGA-II) by Deb et al. (2000) are among the most competitive state of the art MOEAs (Chen et al., 2010). Both of these two MOEAs use some common features: individuals are evaluated under the concept of Pareto dominance, as defined in Goldberg (1989); and elitism is implemented by keeping potentially non-dominated individuals found so far during the genetic search in an external archive, in the case of Zitzler et al., or in the population of the next generation, in the case of Deb et al. Different algorithms have been developed, namely the MOEA/D in Zhang and Li (2007) which uses a strategy of decomposing the multi-objective problem into a number of single objective optimization problems.

In parallel, the evolutionary global search for good fitted solutions within populations has been fruitfully combined with local search, thus producing refinements on individual fitness, within the so-called memetic algorithms. In the sequel, memetic MOEAs have been proposed (Knowles and Corne, 2005). Memetic algorithms have been applied in some scheduling contexts, for single objective (Beddoe et al., 2009; Bai et al., 2010; Burke and Landa Silva, 2005), as well as for multi-objective optimization (Burke et al., 2010).

3.2. *Evolutionary heuristics for the BRP*

In our previous approach to the BRP (Moz et al., 2009), two algorithms were developed: a straightforward adaptation of SPEA2 (there referred to as ASP) and a utopian genetic heuristic (there referred to as UGH). These algorithms relied on an encoding/decoding of each individual using two chromosomes, one representing a permutation of duties and the other a permutation of drivers. BRP solutions were decoded and evaluated by a specially devised constructive heuristic mapping pairs of chromosomes into rosters. The comparison of these two algorithms revealed that the adapted SPEA2 algorithm has shown the ability to attain a better coverage of the Pareto front, while the utopian evolutionary heuristic reached more diversity of potentially non-dominated solutions. These findings motivated the authors to develop a new evolutionary algorithm combining the features of both approaches that contributed to better quality approximated fronts, namely:

(1) the use of a fitness function based on dominance Pareto strength;
(2) the use of an archive to keep a record of the non-dominated individuals within the population, that is, the potentially non-dominated individuals;
(3) the enforcement into the population of the utopian individuals, as well as of the potentially lexicographic individuals.

In addition, an encoding using a single chromosome was adopted, with the purpose of reducing computational effort. This new algorithm was further enhanced by applying a local search procedure that looks for possible improvements on the set of potentially non-dominated individuals. The resulting Memetic Adapted Strength Pareto Based Algorithm (MASP) will be presented in the next section.

4. The memetic adapted strength Pareto based algorithm

4.1. Main algorithm

As mentioned above, MASP applies a local search heuristic within an evolutionary scheme that combines utopian/lexicographic elitism with a strength Pareto fitness evaluation. The algorithm was developed on the basis of SPEA2. In addition to the elitism provided by archiving the most fitted individuals found during the genetic search, potentially lexicographic and utopian individuals are included in the archive and can be selected for variation.

Each utopian individual, ut_1 or ut_2, is associated with a solution obtained by optimizing the corresponding objective, *Objective 1* (f_1) or *Objective 2* (f_2), but relaxing some constraints. Consequently, ut_1 and ut_2 are probably infeasible. Nevertheless they are kept in the archive, while they are not dominated, so as to attract individuals to promising regions in the objective space. Each potentially lexicographic individual, lex_1 or lex_2, is associated with a feasible solution that is, among all the already known solutions, the most suited to optimize the corresponding objective. It is worth noting that the potentially lexicographic individuals correspond to the outer points of the approximated Pareto front in the objective space. Thus, the initial potentially lexicographic individuals are added to the archive of the first generation, but they may be updated during the genetic search.

For illustrating purposes, Fig. 2 displays an example of distribution in the objective space of the individuals from a generation, disclosing the approximated Pareto front, the potentially lexicographic points and the utopian points.

The genetic components include, in each generation, a population of individuals, *Pop*, with a fixed dimension |*Pop*|, and an archive, *Arc*, with a fixed dimension |*Arc*|. Each individual is characterized by a chromosome that represents the list of duties and is coded through an integer vector. Note that this encoding is indirect insofar as we must apply a constructive heuristic (*Generator*) to obtain a solution for BRP, a roster, as detailed in section 4.2. The main steps of MASP are described in Fig. 3.

The initial potentially lexicographic and the utopian individuals are obtained by running a single objective evolutionary procedure (Step 1.1 of MASP) to be described in section 4.3. These four individuals are used to initialize the archive (Step 1.2). The initial population (Step 1.3) is filled with |*Pop*|/2 randomly generated individuals. The remaining |*Pop*|/2 individuals are obtained by

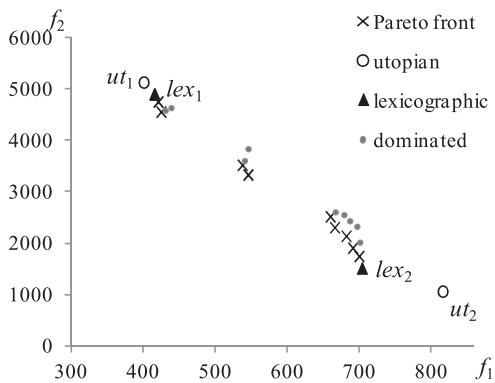

Fig. 2. Individuals from a generation represented in the objective space.

ALGORITHM *MASP*

STEP 1. {initialization}
1.1 CALL *SingleObjectiveEA*(*version*) to compute lex_1, lex_2, ut_1 and ut_2
1.2 $Arc(1) = \{lex_1, lex_2, ut_1, ut_2\}$ {initialize the archive}
1.3 Create the initial population *Pop*(1)
1.4 $t = 1$ {generation counter}
1.5 Set $obj = 3$ {configures *Generator* to evaluate both objective values}

STEP 2. {evaluation, local search and selection}
2.1 FOR $i \in Pop(t)$ DO {for all individuals in *Pop*(*t*)}
CALL *Generator*(*false*,*obj*,*i*) to compute objective values for *i*
2.2 FOR $i \in Pop(t) \cup Arc(t)$ DO
Compute $F(i)$
2.3 Build $Arc(t+1)$ with the best fitted $|Arc|$ individuals from $Pop(t) \cup Arc(t)$
2.4 IF $t \equiv 0 \bmod (NLocal)$ THEN CALL *LocalSearch*($Arc(t+1)$)
2.5 Build a mating pool by selecting $|Pop|$ individuals from
$Pop(t) \cup Arc(t+1)$ using binary tournament with replacement

STEP 3. {recombination and mutation}
3.1 Perform crossover over the individuals in the mating pool and submit them to mutation
3.2 Update *Pop*(*t*+1)
3.3 $t = t+1$

STEP 4. {stopping criterion}
IF $t \leq NGER$ THEN GO TO STEP 2

STEP 5. {solution}
Output = set of non-dominated individuals in $Arc(t+1) \setminus \{ut_1, ut_2\}$

STOP

Fig. 3. Description of the *MASP* algorithm.

perturbing the potentially lexicographic individuals in the following way: $|Pop|/4$ individuals result from swapping two duties randomly selected in lex_1 and the other $|Pop|/4$ individuals result from swapping two duties randomly selected in lex_2.

Then MASP proceeds to the iterative genetic search. In Step 2.1, the *Generator* procedure, presented in detail in section 4.2, transforms each chromosome (individual) into a BRP solution – a roster, allowing for its evaluation concerning the two objective function values. Relying on these values, individuals in $Arc \cup Pop$ are classified as dominated or potentially non-dominated. Then, based on the concept of Pareto strength dominance, proposed in Zitzler et al. (2002), individuals are evaluated in Step 2.2 by a fitness function that measures the "pressure" each individual suffers in terms of dominance by the remaining ones, both in the population and archive, as described in section 4.2. For this reason the fitness function is to be minimized.

The archive of the next generation is built with the best fitted $|Arc|$ individuals (Step 2.3) from $Arc \cup Pop$. If the number of potentially non-dominated individuals, *ndi*, is greater than $|Arc|$, then $ndi - |Arc|$ individuals are randomly selected for discarding. In Step 2.4 of MASP a specially devised bi-objective local search procedure is executed over the archive's new individuals every

NLocal generations, i.e., every t generation with $t \equiv 0 \bmod (NLocal)$. This procedure is described in section 4.4. In Step 2.5, a mating pool is built by the selection operator which uses binary tournament with replacement and considers the individuals in the union of the archive with the population. Then, in Step 3, a new population is obtained through variation on the individuals of the mating pool. Recombination is performed by applying the order crossover operator on pairs of randomly selected individuals, referred to as OX in Michalewicz (1999). The mutation operator acts, with fixed probability, on each individual by swapping the position of two randomly chosen alleles.

The algorithm stops when the number of generations performed, t, attains the maximum number of generations, *NGER* (Step 4).

4.2. Roster building and fitness calculations

The *Generator* algorithm in Fig. 4 builds a BRP solution from a list of duties representing an individual, thus allowing for its evaluation. It is called in Step 2.1 of MASP and is also used within the *SingleObjectiveEA* procedure in Step 1.1 of MASP. The *Generator* procedure is a constructive heuristic which, for each duty in the list of duties, searches for a free driver in the corresponding day to assign the duty. The algorithm considers the list of drivers sorted in non-decreasing order of their overtime cost—Step 1.2. This ordering promotes the assignment of cheaper drivers. However, with probability *PRandom*, drivers are randomly ordered, to avoid potential elimination of part of the solution space—Step 1.3.

In Step 2, the algorithm *Generator* uses: a function *day* that maps each duty to its corresponding day in the rostering period; a function *driverWorkload* that calculates the workload already assigned to a given driver; and a function *driverOvertime* that computes the total overtime already assigned to a given driver in the current solution. A boolean function, *canAssign*(j, i, *rel*), evaluates if driver j can be assigned to duty i. Here, the parameter *rel* indicates whether all the constraints should be considered (*rel* = false) or not (*rel* = true). In the last case, some constraints are relaxed to determine the utopian points, as will be explained in section 4.3. For the case *rel* = false, procedure *Generator* tries to impose satisfaction of all BRP constraints. The context of application of the procedure is given by another parameter, *obj*, which indicates whether *Generator* is called within a single objective optimization of f_1 or f_2 ($obj = 1$ or $obj = 2$, respectively), or else it is called within MASP, to evaluate both objective values ($obj = 3$).

Since *Generator* does not ensure feasibility (even if *rel* = *false*), rosters that do not satisfy all the hard constraints are penalized in both objective values and, as a result, these individuals are less prone to be chosen for reproduction in Step 2.5 of the algorithm MASP. The penalization value – a big non-negative real value – is the same for all the situations where infeasibility occurs, without differentiating violated constraints or the amount of violation. The computational complexity of this procedure is $O(|T||V|)$, since per each duty the algorithm searches for a driver allowing for a feasible assignment.

Returning to MASP, to calculate the fitness of the individual i, represented by $F(i)$, in Step 2.2 of Fig. 3, the process begins by determining the strength of individual i, $S(i)$, which is given by the number of individuals, both in the population and in the archive that are dominated by i. Since the utopian individuals may dominate all the non-dominated individuals that potentially define real

ALGORITHM *Generator*(*rel, obj, individual*)

STEP 1. {initialization}
1.1 Set the list of *duties* equal to the chromosome
1.2 Set the list of *drivers* equal to [1…|*M*|], where drivers are sorted in non-decreasing order of overtime cost
1.3 With probability *PRandom* shuffle the list of *drivers* randomly
1.4 All drivers are set free for all days and all duties are set unassigned

STEP 2. {assignments}
2.1 Set $i = 1$ {duty index}
continue=true {it is possible to continue the assignment}
2.2 WHILE ($i \leq |T|$ and *continue*) {search for an assignment}
IF $obj = 1$ THEN
Search for j such that *canAssign*(*drivers*[*j*], *duties*[*i*], *rel*) AND
$\forall k$: *canAssign*(*drivers*[*k*], *duties*[*i*], *rel*) $\Rightarrow$ *driverWorkload*(*drivers*[*k*]) $\geq$ *driverWorkload*(*drivers*[*j*])
{*j* has minimum workload}
ELSE IF $obj = 2$ THEN
Search for j such that *canAssign*(*drivers*[*j*], *duties*[*i*], *rel*) AND
$\forall k$: *canAssign*(*drivers*[*k*], *duties*[*i*], *rel*) $\Rightarrow$ *driverOvertime*(*drivers*[*k*]) $\geq$ *driverOvertime*(*drivers*[*j*])
{*j* has minimum overtime}
ELSE
Search for *j* such that *canAssign*(*drivers*[*j*], *duties*[*i*], *rel*)
IF *j* is found THEN {assignment}
Assign *duties*[*i*] to *drivers*[*j*], set *duties*[*i*] as assigned and *drivers*[*j*] as non-free in *day*(*duties*[*i*])
ELSE *continue*=false
i=*i*+1

STEP 3. {stopping criterion}
IF all duties are assigned THEN {a roster is found}
Compute the objective value(s) associated with the roster
ELSE {the solution is infeasible}
Assign an adequate penalty to objective function(s)

STOP

Fig. 4. Description of the *Generator* algorithm.

Pareto front points, they are not considered for the strength computation and their fitness values are set to zero. Hence,

$$S(i) = |\{j | j \in Pop \cup Arc \backslash \{ut_1, ut_2\}, i \succ j\}|,$$

where $i \succ j$ means that individual i dominates j, that is, the roster built from i performs better than the roster built from j on at least one of the objectives and does not perform worse on the other. Then, the fitness of an individual i, $F(i)$, is given by the total strength of individuals dominating i plus an estimate for the diversity whose aim is to penalize individuals in more crowded regions of the objective space:

$$F(i) = \sum_{j \in Pop \cup Arc \backslash \{ut_1, ut_2\}, j \succ i} S(j) + \frac{1}{\sigma_i^k + 2}.$$

ALGORITHM *SingleObjectiveEA* (*version*)

STEP 1. {initialization}
1.1 Create the initial population $Pop(1)$
1.2 Set *best* as a randomly chosen individual
1.3 Set *rel* according to *version*
1.4 IF fitness function = f_1 THEN *obj*=1 ELSE *obj*=2
1.5 $t = 1$ {generation number}

STEP 2. {evaluation and selection}
2.1 FOR all $i \in Pop(t)$ DO
CALL *Generator*(*rel*, *obj*, *i*) to compute objective value(s) for *i*
FOR all $i \in Pop(t)$ DO
Calculate $F(i)$ according to *version*
2.2 Set $i_{\min} = \arg \min_{i \in Pop(t)} F(i)$
IF $F(i_{\min}) < F(best)$ THEN $best = i_{\min}$
2.3 Build a mating pool by selecting $|Pop|-1$ individuals from $Pop(t)$
2.4 Insert *best* in the mating pool

STEP 3. {recombination and mutation}
3.1 Perform crossover over individuals in the mating pool and submit them to mutation
3.2 Update $Pop(t+1)$
3.3 $t = t+1$

STEP 4. {stopping criterion}
4.1 IF $t \leq NGERsi$ THEN GO TO STEP 2

STEP 5. {solution}
Output = *best*

STOP

Fig. 5. Description of the single objective evolutionary algorithm.

The diversity component is expressed by $1/(\sigma_i^k + 2)$, where σ_i^k is the Euclidean distance between i and its k^{th} nearest neighbor in the objective space with $k = \lfloor\sqrt{|Pop| + |Arc|}\rfloor$, the integer part of the square root of the total number of individuals in the current generation (Zitzler et al., 2002). An individual not dominated by any other within the current archive or population (except, eventually, by the utopians) is assigned a fitness value below 1.

4.3. Computing the initial potentially lexicographic and the utopian individuals

The single objective evolutionary algorithm presented in Fig. 5 runs four times in Step 1.1 of MASP to compute both the utopian and the initial potentially lexicographic individuals: ut_1, ut_2 and lex_1, lex_2, respectively. The search for each of these individuals is oriented by the respective fitness function. As in the main components of MASP, this algorithm also considers a fixed dimension population of $|Pop|$ individuals, all coded in the same way. Elitism is implemented by keeping a record of the best individual found so far, which is inherited from the previous population and

Table 1
Versions for the single objective evolutionary algorithm

Feature\version	LEX1	LEX2	UT1	UT2
Fitness function $F(i)$	f_1 (followed by $\min f_2$ s.t. $f_1 \leq f_1^{\min}$)	f_2 (followed by $\min f_1$ s.t. $f_2 \leq f_2^{\min}$)	f_1	f_2
rel	false	false	true	true
Initialization of *best*	Randomly chosen individual from *Pop*(1)			
Output for MASP	lex_1	lex_2	ut_1	ut_2

updated in every generation. The variation operators at Step 3 are the same as those used in MASP: selection by binary tournament with reposition, crossover OX for recombination and randomly swapping two alleles' positions for mutation.

The algorithm *SingleObjectiveEA* receives as input a parameter – *version* – that stipulates the goal of the run. Table 1 displays, for each *version* value: the fitness function to be used; the way *Generator* will enforce BRP constraints, given by *rel*; the initialization of variable *best*; and finally, the output of the algorithm.

Considering the determination of $ut_1(ut_2)$, *version* assumes value UT1 (UT2), and the objective function f_1 (f_2) is used as the fitness function. For initialization, *best* individual in Step 1.2 is a randomly chosen individual from the initial population *Pop*(1). The value of parameter *rel* is set to *true* when the aim is to find a solution with a minimum value of f_1 (f_2) by relaxing some constraints, thus possibly leading to an infeasible solution. Step 2 computes fitness values for all individuals in the population and updates the best found so far, if such is the case, while Step 3 performs variation. After *NGERsi* iterations (checked in Step 4), at Step 5, the algorithm outputs the solution corresponding to *best* that is $ut_1(ut_2)$.

For the determination of lex_1 (lex_2), *version* assumes value LEX1 (LEX2), and the objective function f_1 (f_2) is used as the fitness function. For initialization, *best* individual in Step 1.2 is a randomly chosen individual from the initial population *Pop*(1). The value of parameter *rel* is set to *false* as the aim is to find a feasible solution. Step 2 computes fitness values for all individuals in the population and updates the best found so far, if such is the case. After minimizing f_1 (f_2), the *SingleObjectiveEA* runs again to search for a better fitted individual as regards f_2 (f_1) without worsening the value of f_1 (f_2) already attained, $f_1^{\min}$ ($f_2^{\min}$). At Step 5, the solution corresponding to *best* is $lex_1(lex_2)$.

4.4. MASP local search

MASP embeds a local search component to improve the archive with new potentially non-dominated individuals (see Fig. 6). Here, the neighborhood of an individual i, $\upsilon(i)$, is defined as the set of individuals obtained by swapping two drivers' assignments of duties of the same day. Thus, the cardinality of $\upsilon(i)$ is $|\upsilon(i)| = 28\lceil\frac{|V|(|V|-1)}{2}\rceil$, where $\lceil x\rceil$ represents the smallest integer not greater than x. The algorithm explores a random number of individuals, in the neighborhood of i, bounded by $28maxNeighbors$.

The local search strategy consists of updating the approximated Pareto front in *Arc*, in one of two ways:

```
ALGORITHM LocalSearch(Arc)
t = 0
REPEAT {tries to improve each new individual in Arc}
    1.1 Randomly select a new individual i ∈ Arc
        t = t + 1
    1.2 Randomly generate maxNeighbors
        FOR count = 1,…,maxNeighbors DO
            Randomly generate a pair {j, k} of drivers not yet checked
            FOR all day = 1,…,28 DO
                Build neighbor by swapping the duties assigned to j and k on day day
                IF neighbor is feasible and dominates i
                THEN
                    i=neighbor
                    Eliminate from Arc all the individuals dominated by neighbor
                    t = t+1
                    GO TO 1.2
                IF neighbor is feasible and equivalent to i and |Arc| − ndi > 0
                THEN
                    Add neighbor to Arc
                    Eliminate from Arc all the individuals dominated by neighbor
UNTIL all new individuals in Arc are "neighborhood" explored OR t = NIter

Output = Arc
STOP
```

Fig. 6. Description of the *LocalSearch* algorithm.

(1) substituting individual i by a neighbor, if it generates a feasible solution that dominates i and resuming the procedure from the new individual onwards;
(2) inserting in *Arc* a neighbor of individual i, if this new individual corresponds to a potentially non-dominated feasible solution (meaning that it is equivalent to i) and $|Arc| - ndi > 0$.

The search stops when a given number of new individuals explored, *Niter*, is attained, or all the new individuals in *Arc* are explored. The computational complexity of this procedure is $O(|V|^2)$.

The next section describes the computational experiments and presents their respective results.

5. Computational tests

The experiments performed set out to evaluate the behavior of MASP, assessing the individual contribution of its specific features. For this purpose, we ran two simpler versions of the algorithm each of which differing from MASP by sequentially disabling the following two components: local search and utopian/lexicographic elitism.

Table 2 displays a summary of MASP's main features in comparison with those of WASP, the version in which local search was suppressed, and those of ASP, the version without local search and utopian/lexicographic elitism.

Table 2
Features of the genetic algorithms under evaluation

Feature\algorithm	MASP	WASP	ASP
Individual encoding	One chromosome	One chromosome	One chromosome
Fitness function	Strength Pareto based	Strength Pareto based	Strength Pareto based
Archive use	Fixed dimension	Fixed dimension	Fixed dimension
Local search	Yes	No	No
Potentially lexicographic individuals	One for each objective	One for each objective	No
Utopian individuals	One for each objective	One for each objective	No

5.1. Description of the instances

Computational tests with the evolutionary algorithms were performed over data for rostering obtained from benchmark random instances of the Integrated Multi-depot Vehicle and Crew Scheduling Problem (VCRP), provided by Huisman et al. (2005). These VCRP instances were solved in Mesquita et al. (2009). The first set, P80, includes 10 BRP instances resulting from the solution of 80 trips VCRP instances. These BRP instances include 467.2 duties on average for the rostering period. Note that the set of duties is repeated from Monday to Friday, and is reduced for the weekend days. The 10 instances in the second set, P100, came from solutions of 100 trips VCRP instances and on average comprise 616.8 duties. Finally, the third set, Pc, includes 12 BRP instances based on real data from a Portuguese urban bus company under study (Mesquita et al., 2011). In this set, the average number of duties is 856.3.

According to the description of the problem in section 2, each instance of BRP is defined by the following data: V, nd – pool of drivers and number of drivers in the fixed working pool, respectively; T_h ($h = -5, \ldots, -1, 0, 1, \ldots, 28$); T_{ih} ($h = 1, \ldots, 28$; $i \in T_{h-1}$); t_{ih}($h = 1, \ldots, 28$; $i \in T_{h-1}$); e^v_{ih} ($v \in V$; $i \in T_h$; $h = -5, \ldots, -1, 0$); c^v ($v \in V$); c^S; $g = 6$; $\bar{t} = 8$; $b_1 = 48$; $b_2 = 176$; $d_w = 2$; $d_s = 1$.

Though there are no explicit shifts, duties are classified into two groups: early duties starting between 6:00 a.m. and 3:30 p.m., and late duties starting between 3:30 p.m. and 12:00 p.m. According to constraints (h3), sets T_{ih} include all late duties for day h, if i is a late duty on day h-1; otherwise, they include all duties (early or late) for day h. Table 3 displays for each instance its name in column 1, the total number of duties in the rostering period in column 2, the number of early duties in week/weekend days in column 3 and the number of late duties in week/weekend days in column 4.

5.2. Implementation details

As for the parameter values, for the evolutionary algorithms, MASP, WASP, ASP and *SingleObjectiveEA*, $|Pop| = 40$, the probability of crossover was set to 0.8, while the mutation probability was set to 0.2, and $NGER = 2000$ was considered as the stopping criterion. Each initial lexicographic and utopian individual is achieved by performing $NGERsi = 2000$ generations in algorithm *SingleObjectiveEA*. For the three bi-objective algorithms, $|Arc| = 40$. In all the experiments of

Table 3
Features of the instances

1 Instance	2 *NDuties*	3 *NEarlies* Week/weekend day	4 *NLates* Week/weekend day
P80_1	456	10/4	8/8
P80_2	352	7/2	7/7
P80_3	472	11/7	7/7
P80_4	456	11/5	7/7
P80_5	444	9/5	8/8
P80_6	416	10/6	6/6
P80_7	476	11/4	8/8
P80_8	568	13/7	9/9
P80_9	440	10/2	8/8
P80_10	592	13/3	11/11
Average	*467.2*	*10.5/4.5*	*7.9/7.9*
P100_1	720	16/10	12/10
P100_2	628	14/6	11/10
P100_3	648	14/8	12/8
P100_4	572	12/5	11/9
P100_5	648	15/7	11/9
P100_6	516	10/3	11/9
P100_7	624	17/7	9/6
P100_8	656	13/7	13/10
P100_9	704	17/7	11/11
P100_10	452	12/2	7/7
Average	*616.8*	*14.0/6.2*	*10.8/8.9*
Pc122–1	428	11/5	6/6
Pc122–2	428	11/5	6/6
Pc122–3	428	11/5	6/6
Pc224–1	964	27/6	14/12
Pc224–2	980	27/8	14/12
Pc224–3	972	26/6	15/13
Pc226–1	824	22/6	12/12
Pc226–2	852	23/7	12/12
Pc226–3	872	22/5	14/14
Pc238–1	1188	37/18	12/8
Pc238–2	1180	37/15	12/10
Pc238–3	1160	38/17	10/8
Average	*856.3*	*24.3/8.6*	*11.1/9.9*
Total average	*659.9*	*16.8/6.6*	*10.0/9.0*

MASP, $NLocal = 100$, $maxNeighbors \leq |V|$ and $NIter = 10$. The parameter *PRandom* in *Generator* is set to 0.2.

Results for MASP, WASP and ASP were obtained by performing 10 runs of each algorithm and, at the end, by computing the final approximation of the Pareto front by merging the fronts obtained in all the runs and extracting the non-dominated points.

All algorithms were coded in C language and the experiments were carried out on an Intel Core2 processor running at 2.53 GHz and using 4 GB of RAM.

5.3. Performance metrics

In the context of this application, rostering is controlled by human planners, who are interested in examining a variety of good quality solutions. Therefore, the cardinality of the Pareto front approximation is one of the performance metrics used to evaluate the solutions' diversity. This metric, also known as front occupation, was first used in Van Veldhuizen (1999) and indicates the number of potentially non-dominated points obtained from the individuals in the populations (main population and archive) of the last generation. The spread of the Pareto front, given by the distance between the two outer points, here approximated by lex_1 and lex_2, allows one to measure the extent of the front in the objective space.

The individuals generated during the genetic search may correspond to infeasible BRP solutions. Thus, the total number of points corresponding to feasible solutions, feasible points, is calculated.

Another important metric that attempts to highlight if points in the approximated front are evenly distributed (or not) is the spacing metric proposed in Schott (1995): $SM = (\frac{1}{n-1}\sum_{i=1}^{n}(\bar{d} - d_i)^2)^{1/2}$, where n is the number of points in the approximated front, $d_i = \min_j(|f_1^i - f_1^j| + |f_2^i - f_2^j|)$ for i, $j = 1, \ldots, n$, $\bar{d}$ is the mean of all d_i and f_1^i, f_2^i are the coordinates of the ith point in the objective space. The spacing metric divided by the mean gives the coefficient of variation, $CV = SM/\bar{d}$. A value of zero for this metric indicates that all points are equally distributed.

For each of the three sets of instances, the analysis of the results is reported and discussed from two perspectives. Firstly, the metrics were computed to evaluate average results from the 10 runs of each algorithm. In the sequel, the focus goes to the final approximation of the Pareto front (*FAP*) obtained from all the performed runs of each algorithm per instance of the problem. This set is obtained by merging the potentially non-dominated points of all these runs and, afterwards, eliminating all the dominated points. Again, the front occupation of *FAP* will be analyzed as well as the coefficient of variation.

Moreover, a measure of the relative strength of *FAP* is used for pair-wise comparison of any two algorithms. Consider *FAP*1 and *FAP*2 two approximations of the Pareto front. The relative strength of *FAP*1 over *FAP*2 is given by $|\{i \in FAP1 : \exists i' \in FAP2, i \succ i'\}|/|FAP1|$, which counts the number of points in *FAP*1 that dominate at least one point in *FAP*2 over the total number of points in *FAP*1. A similar metric within the solutions' space was proposed in Zitzler and Thiele (1999).

Finally, for each instance, the best known approximation of the Pareto front (*BKAPF*) is built, that is, the set of potentially non-dominated points in the union of the *FAP* produced by two algorithms (*ALG1* and *ALG2*). The contribution of *ALG1* (*ALG2*) to the *BKAPF* is assessed through the proportion of points in *BKAPF* that were obtained from the *FAP* of *ALG1* (*ALG2*).

5.4. Numerical results

Table 4 displays average results obtained per run (of the 10 runs performed) for each test instance of BRP. Its analysis allows one to evaluate a single run of each of the three algorithms. Column 1

Table 4
Average results per run

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
	ASP					WASP						MASP						
1 Instance	*spread*	*fo*	*CV*	*nfp*	cpu	*spread*	*fo*	*CV*	*nfp*	cpu_ init	cpu_ gen	*spread*	*fo*	*CV*	*nfp*	cpu_ init	cpu_ gen	cpu_ ls
P80_1	2737.4	15.4	2.1	70.6	56.3	3267.0	24.7	2.5	78.2	125.7	70.5	3255.8	21.2	2.4	72.4	125.3	76.6	6.0
P80_2	3365.2	15.6	1.9	71.2	45.9	3852.9	26.1	1.7	79.1	101.2	57.0	3507.9	19.7	3.0	74.3	101.4	59.4	2.5
P80_3	3265.4	12.0	1.8	71.4	58.2	3943.5	15.0	2.5	78.5	130.7	72.2	3858.3	13.3	2.2	76.4	131.2	79.3	6.2
P80_4	2816.3	15.3	1.8	71.2	56.8	3458.3	23.0	2.4	78.0	125.9	70.1	3475.2	21.9	2.2	74.2	127.0	77.2	6.7
P80_5	2987.6	15.8	2.1	72.8	55.9	3592.7	24.3	2.9	78.4	123.4	68.8	3725.0	23.8	2.3	71.0	124.0	75.7	6.6
P80_6	3463.0	15.6	2.0	74.3	53.2	3862.3	16.9	1.9	78.6	117.2	65.5	3847.3	17.4	2.5	69.9	117.3	70.2	4.5
P80_7	2525.4	13.7	2.1	71.0	58.6	3311.6	23.8	2.0	78.8	129.4	72.2	3056.7	19.0	2.6	69.6	130.1	79.4	6.6
P80_8	2114.8	14.9	2.0	73.0	68.0	3430.4	28.1	2.1	79.4	150.7	83.7	2997.3	19.6	2.3	69.3	151.0	96.5	12.6
P80_9	2866.9	15.0	1.9	71.1	54.3	3688.3	23.4	1.8	78.3	120.8	68.5	3547.6	19.1	2.1	74.6	121.1	75.0	6.4
P80_10	1335.3	16.8	2.0	71.8	68.8	2438.8	31.7	2.1	78.7	154.5	85.9	2315.0	22.3	1.9	67.8	155.0	101.8	15.4
Average	*2747.7*	*15.0*	***2.0***	*71.8*	*57.6*	***3484.6***	***23.7***	*2.2*	***78.6***	*128.0*	*71.4*	*3358.6*	*19.7*	*2.4*	*72.0*	*128.3*	*79.1*	*7.4*
P100_1	1575.6	15.1	1.9	70.9	109.5	2707.1	25.4	2.0	78.9	255.1	141.3	2464.9	20.5	2.5	69.5	257.5	166.5	25.3
P100_2	2523.2	15.8	2.0	73.5	98.6	3376.5	29.8	2.2	77.8	229.9	124.6	3259.0	23.2	2.9	72.3	230.3	144.4	19.0
P100_3	2823.4	14.8	2.2	69.8	100.5	4298.0	29.5	2.4	78.9	232.6	128.0	4032.5	23.3	2.1	71.7	235.2	153.2	23.6
P100_4	2893.2	13.1	2.0	72.8	91.6	3711.0	25.3	2.4	78.3	210.7	115.8	3627.5	21.2	2.7	73.4	210.9	132.5	15.6
P100_5	2482.8	15.3	2.1	72.9	100.5	3316.0	28.3	2.5	78.4	233.0	128.6	3286.3	23.7	2.3	74.8	233.0	154.3	25.3
P100_6	3064.4	12.7	1.9	72.2	84.3	3046.9	21.5	2.1	79.3	194.1	103.5	2988.1	16.4	2.8	70.1	194.0	110.6	6.4
P100_7	2342.8	14.8	2.0	70.8	96.3	3318.3	29.8	2.4	79.0	222.7	121.3	3244.0	24.4	2.2	71.3	222.4	137.5	16.7
P100_8	2992.1	14.6	1.8	70.5	101.7	4415.9	23.1	2.1	79.1	235.8	130.2	4355.7	19.1	1.9	67.7	236.8	154.4	23.4
P100_9	2031.7	16.2	1.8	72.6	107.4	3089.0	26.4	2.0	78.1	254.8	139.1	3486.9	22.9	2.2	71.1	248.0	157.5	23.8
P100_10	3076.3	10.9	2.3	72.7	75.0	3389.0	26.2	1.8	78.2	181.5	96.8	3299.5	24.1	2.3	73.7	174.5	103.1	8.9
Average	*2580.6*	*14.3*	***2.0***	*71.9*	*96.5*	***3466.8***	***26.5***	*2.2*	***78.6***	*225.0*	*122.9*	*3404.4*	*21.9*	*2.4*	*71.6*	*224.3*	*141.4*	*18.8*
Pc122–1	6291.4	12.7	2.1	71.3	116.9	6744.9	23.7	2.2	78.5	278.9	149.1	6509.7	21.8	1.9	75.2	283.8	162.5	8.2
Pc122–2	6470.7	12.3	2.2	69.9	117.3	6729.4	17.7	1.9	78.0	275.5	149.2	6669.1	15.2	2.7	71.4	282.0	161.5	7.7
Pc122–3	7498.3	12.1	1.9	71.2	117.0	7996.7	18.2	1.8	78.1	277.4	151.5	7981.1	18.8	2.3	73.2	286.2	161.1	7.9
Pc224–1	4322.0	13.5	2.0	71.4	211.7	6443.6	18.0	1.9	77.9	496.5	268.4	6272.8	13.4	2.0	69.4	509.8	342.4	70.4
Pc224–2	3923.9	8.7	1.3	70.7	218.4	6395.9	9.2	1.9	78.6	510.4	277.8	5968.9	10.1	1.7	75.9	522.7	341.1	63.4
Pc224–3	3698.7	13.8	1.7	71.5	213.1	6718.5	19.0	1.8	79.2	499.6	271.1	6242.0	14.0	2.2	75.3	498.2	344.1	74.3
Pc226–1	6464.1	9.9	2.0	71.2	193.0	7953.4	16.2	2.4	78.3	451.6	253.0	7782.8	12.6	2.2	70.4	450.7	291.5	44.7
Pc226–2	5790.7	9.9	1.7	70.7	198.8	8107.3	14.1	2.0	78.3	479.3	258.9	7106.1	9.6	2.1	69.6	462.3	293.3	39.8
Pc226–3	5713.4	11.5	2.5	70.6	203.4	7122.8	12.0	1.6	77.9	469.8	253.9	6901.6	9.1	1.8	68.9	468.7	297.6	43.6
Pc238–1	6397.3	10.1	1.8	70.1	407.8	9643.8	18.2	2.1	78.1	953.5	522.0	8940.5	13.2	2.1	72.5	960.3	717.1	185.5
Pc238–2	6546.5	10.4	1.7	72.2	404.2	9160.2	18.6	2.1	78.8	952.8	520.8	9274.4	15.0	2.2	69.1	954.0	703.4	178.2
Pc238–3	6873.6	11.5	1.9	71.0	399.9	8975.0	17.4	2.5	78.7	943.2	509.9	8944.9	14.5	2.2	70.7	958.9	707.3	186.5
Average	*5832.6*	*11.4*	***1.9***	*71.0*	*233.5*	***7666.0***	***16.9***	*2.0*	***78.4***	*549.0*	*298.8*	*7382.8*	*13.9*	*2.1*	*71.8*	*553.1*	*376.9*	*75.9*
Total average	*3852.3*	*13.4*	***2.0***	*71.5*	*135.7*	***5047.0***	***22.0***	*2.1*	***78.5***	*316.2*	*172.8*	*4882.0*	*18.2*	*2.3*	*71.8*	*317.6*	*210.3*	*36.6*
Improvement over ASP						***31.0***	***64.2***	*–5.0*	***9.8***			*26.7*	*35.8*	*–15.0*	*0.3*			

shows the names of the instances. Columns 2–6 contain the figures for metrics and computational time of ASP, while columns 7–12 refer to WASP and columns 13–19 to MASP. For each algorithm, the corresponding figures are displayed under the following column headers: *spread* (front spread); *fo* (front occupation); *CV* (coefficient of variation); *nfp* (number of feasible points). The remaining columns display CPU time in seconds. Column 6 refers to the total CPU for ASP. Columns 11 and

17 refer to the CPU time needed to approximate the initial potentially lexicographic and utopian solutions for WASP and MASP, respectively. Columns 12 and 18 exhibit the computing time for WASP and MASP, respectively, excluding the initialization (Step 1.1 in Fig. 3). Finally, column 19 highlights the amount of CPU time spent by the *LocalSearch* procedure (this time is part of the total time given in column 18). The last line of Table 4 presents the percentual relative improvement of WASP and MASP over ASP for all metrics.

The best features for all metrics were highlighted in bold for the average results per group of instances and also for the average results of all instances.

Our analysis of Table 4 focuses on evaluating the effect of the two enhancement components over the basic version of the algorithm ASP: utopian/lexicographic elitism included in WASP and MASP, and the local search in MASP.

The average results of each group of instances show that WASP and MASP clearly outperform ASP, except with respect to *CV* which is slightly worst, as shown by columns 4, 9 and 15.

A single run of WASP on average outperforms a single run of MASP, except obviously as regards the computational time, as MASP additionally performs the local search procedure. The time required by the local search procedure *LocalSearch* was relatively small, as can be perceived by column 19.

Table 5 displays the results for the final approximations of the Pareto front. For each instance, the approximations obtained by 10 independent runs of each algorithm are merged and a final approximation of the Pareto front (*FAP*) is obtained by eliminating the dominated points. The absolute metrics are presented in columns 2–7 and the pair-wise comparison metrics are presented in the remaining columns. The columns in Table 5 show, respectively, the front occupation (#*FAP*), in columns 2, 4 and 6; and the coefficient of variation (*CV*), in columns 3, 5 and 7. Columns 8–9 display the comparison of ASP against WASP, while columns 10–11 display the opposite comparison, that is, WASP against ASP. Columns 12–13 and 14–15 display the corresponding comparison for ASP and MASP, while columns 16–17 and 18–19 present the pair-wise comparisons of WASP and MASP. The percentual relative contribution to FAP (*ctrFAP*) and the percentual relative strength (*RStr*) are displayed in columns 8, 10, 12, 14, 16 and 18, and in columns 9, 11, 13, 15, 17 and 19, respectively.

Analyzing the figures in Table 5 we can conclude that MASP clearly outperforms WASP, which confirms the relevance of the local search procedure (columns 16–19). Concerning the front occupation (columns 2, 4 and 6), MASP attained on average much better figures than WASP. As for the coefficient of variation (columns 3, 5 and 7), WASP performed slightly better. As to the best known approximated Pareto front, once again, MASP showed its superiority, contributing with more than 90% of points for 13 over 32 instances, completely filling five of them (column 18). As for the relative strength, MASP attained better values than WASP, reaching 78.9%, 82.6% and 82.2%, on average, over WASP, respectively, for sets P80, P100 and Pc (column 19).

The pair-wise comparison of WASP with ASP (columns 8–11) reveals that WASP clearly outperforms ASP with respect to the contribution for the final Pareto front and it has a similar behavior regarding the relative strength, thus revealing the contribution of the utopian/lexicographic elitism.

By comparing MASP versus ASP (columns 12–15), we can conclude that MASP has a much better performance in both metrics. It is worth noting that MASP totally filled the *FAP* for nine over the 32 instances. In the group of Pc instances, which are the most difficult, this happened for eight of the 12 instances. Additionally, it contributed with more than 90% of points in *FAP* for 14 instances.

Table 5
Results from the final approximations of the Pareto front

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
	ASP		WASP		MASP		ASP vs. WASP		WASP vs. ASP		ASP vs. MASP		MASP vs. ASP		WASP vs. MASP		MASP vs. WASP	
Instance	# *FAP*	*CV*	# *FAP*	*CV*	# *FAP*	*CV*	*ctrFAP* (%)	*RStr* (%)	*ctrFAP* (%)	*RStr* (%)	*ctrFAP* (%)	*RStr* (%)	*ctrFAP* (%)	*RStr* (%)	*ctrFAP* (%)	*RStr* (%)	*ctrFAP* (%)	*RStr* (%)
P80_1	17	1.7	37	2.6	40	3.0	38.2	70.6	61.8	16.2	17.5	35.3	82.5	45.0	16.7	16.2	83.3	77.5
P80_2	16	2.0	26	1.2	29	1.3	35.7	37.5	64.3	15.4	24.2	25.0	75.8	34.5	12.5	3.8	87.5	75.9
P80_3	13	2.5	24	0.9	21	1.0	4.2	7.7	95.8	75.0	0.0	0.0	100.0	90.5	23.8	16.7	76.2	66.7
P80_4	20	1.2	29	2.4	32	2.2	48.5	60.0	51.5	17.2	25.0	25.0	75.0	40.6	22.9	17.2	77.1	75.0
P80_5	21	1.6	28	0.9	42	1.0	52.8	61.9	47.2	10.7	25.5	33.3	74.5	33.3	7.3	7.1	92.7	78.6
P80_6	17	1.5	12	0.5	36	0.8	57.1	29.4	42.9	25.0	5.7	11.8	94.3	61.1	13.3	25.0	86.7	41.7
P80_7	15	1.2	26	0.8	39	0.9	34.6	60.0	65.4	15.4	10.0	26.7	90.0	43.6	7.7	3.8	92.3	87.2
P80_8	23	1.7	32	0.9	34	1.7	40.0	56.5	60.0	34.4	10.8	4.3	89.2	64.7	8.3	3.1	91.7	97.1
P80_9	26	2.7	26	0.7	31	1.2	65.5	69.2	34.5	19.2	32.4	30.8	67.6	64.5	0.0	0.0	100.0	96.8
P80_10	20	1.4	39	1.1	38	1.1	41.9	75.0	58.1	7.7	20.9	25.0	79.1	39.5	0.0	0.0	100.0	92.1
Average	*18.8*	*1.7*	*27.9*	***1.2***	***34.2***	*1.4*	*41.9*	***52.8***	***58.2***	*23.6*	*17.2*	*21.7*	***82.8***	***51.7***	*11.3*	*9.3*	***88.8***	***78.9***
P100_1	20	1.3	32	1.1	23	0.8	40.5	65.0	59.5	21.9	12.5	10.0	87.5	60.9	0.0	0.0	100.0	95.7
P100_2	18	1.0	38	0.8	35	1.6	13.9	27.8	86.1	26.3	10.5	5.6	89.5	48.6	13.5	7.9	86.5	82.9
P100_3	15	1.3	35	1.1	29	1.1	21.6	53.3	78.4	45.7	7.4	13.3	92.6	51.7	12.1	0.0	87.9	100.0
P100_4	18	1.1	31	1.2	32	1.3	38.2	72.2	61.8	22.6	8.6	0.0	91.4	59.4	3.1	3.2	96.9	93.8
P100_5	18	1.6	31	1.9	39	2.4	31.4	55.6	68.6	19.4	16.3	22.2	83.7	35.9	16.7	16.1	83.3	66.7
P100_6	19	1.5	26	0.8	21	1.0	33.3	47.4	66.7	11.5	4.5	0.0	95.5	42.9	11.1	3.8	88.9	61.9

Table 5
Continued

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
	ASP		WASP		MASP		ASP vs. WASP		WASP vs. ASP		ASP vs. MASP		MASP vs. ASP		WASP vs. MASP		MASP vs. WASP	
Instance	# *FAP*	*CV*	# *FAP*	*CV*	# *FAP*	*CV*	*ctrFAP* (%)	*RStr* (%)	*ctrFAP* (%)	*RStr* (%)	*ctrFAP* (%)	*RStr* (%)	*ctrFAP* (%)	*RStr* (%)	*ctrFAP* (%)	*RStr* (%)	*ctrFAP* (%)	*RStr* (%)
P100_7	19	0.8	38	1.8	36	1.5	22.5	36.8	77.5	18.4	10.8	5.3	89.2	41.7	17.9	10.5	82.1	80.6
P100_8	17	1.5	27	1.7	33	2.3	46.4	52.9	53.6	33.3	14.3	23.5	85.7	63.6	15.8	3.7	84.2	87.9
P100_9	18	1.0	37	0.6	39	2.4	18.9	38.9	81.1	35.1	4.9	0.0	95.1	56.4	5.0	2.7	95.0	92.3
P100_10	16	2.1	38	2.5	36	2.5	30.3	62.5	69.7	23.7	21.1	25.0	78.9	47.2	29.4	18.4	70.6	63.9
Average	*17.8*	***1.3***	***33.3***	***1.3***	*32.3*	*1.7*	*29.7*	***51.2***	***70.3***	*25.8*	*11.1*	*10.5*	***88.9***	***50.8***	*12.5*	*6.6*	***87.5***	***82.6***
Pc122–1	19	1.7	31	1.3	26	1.7	31.0	36.8	69.0	51.6	14.8	21.1	85.2	76.9	35.7	29.0	64.3	53.8
Pc122–2	15	2.1	29	1.1	25	1.3	31.0	33.3	69.0	51.7	14.8	13.3	85.2	72.0	21.4	13.8	78.6	84.0
Pc122–3	17	2.3	23	0.9	27	1.0	28.0	35.3	72.0	65.2	14.8	23.5	85.2	66.7	22.2	21.7	77.8	66.7
Pc224–1	15	0.9	19	1.3	21	1.3	21.7	20.0	78.3	63.2	0.0	0.0	100.0	81.0	9.1	5.3	90.9	90.5
Pc224–2	12	0.9	15	2.1	15	1.4	6.7	8.3	93.3	93.3	0.0	0.0	100.0	100.0	18.8	13.3	81.3	66.7
Pc224–3	18	2.0	24	1.4	20	1.2	13.0	16.7	87.0	70.8	0.0	0.0	100.0	95.0	9.1	0.0	90.9	100.0
Pc226–1	12	1.0	20	1.4	16	1.9	0.0	0.0	100.0	80.0	0.0	0.0	100.0	93.8	5.9	0.0	94.1	100.0
Pc226–2	13	1.2	17	0.9	14	0.9	0.0	0.0	100.0	82.4	0.0	0.0	100.0	100.0	0.0	0.0	100.0	100.0
Pc226–3	16	0.7	15	2.0	14	1.3	25.0	18.8	75.0	73.3	0.0	0.0	100.0	100.0	0.0	0.0	100.0	100.0
Pc238–1	15	1.4	20	1.0	19	1.9	15.0	20.0	85.0	70.0	0.0	0.0	100.0	94.7	18.2	5.0	81.8	94.7
Pc238–2	12	1.7	21	1.5	20	1.4	17.4	25.0	82.6	85.7	0.0	0.0	100.0	95.0	45.5	42.9	54.5	55.0
Pc238–3	16	2.1	24	1.6	16	1.4	18.5	25.0	81.5	58.3	16.7	6.3	83.3	62.5	45.5	37.5	54.5	75.0
Average	*15*	*1.5*	***21.5***	***1.4***	*19.4*	*1.4*	*17.3*	*19.9*	***82.7***	***70.5***	*5.1*	*5.4*	***94.9***	***86.5***	*19.3*	*14.0*	***80.7***	***82.2***
Total average	*17.1*	*1.5*	*27.2*	***1.3***	***28.1***	*1.5*	*28.8*	*40.0*	***71.2***	***41.9***	*10.8*	*12.1*	***89.3***	***64.5***	*14.6*	*10.0*	***85.4***	***81.3***

Table 6
Comparison of the lexicographic solutions approximated by MASP and the ones achieved by CPLEX solver

1	2	3	4	5	6	7
	lex_1			lex_2		
Instance	Δf_1	Δf_2	Dominance relation	Δf_1	Δf_2	Dominance relation
P80_1	0.48	− 3.47	$\sim$	1.44	11.28	$\prec$
P80_2	1.24	− 1.72	$\sim$	1.01	1.18	$\prec$
P80_3	1.76	− 2.49	$\sim$	0.14	14.23	$\prec$
P80_4	0.00	− 2.10	$\sim$	0.86	6.27	$\prec$
P80_5	1.05(*)	− 2.14(*)	$\sim$	1.59	7.31	$\prec$
P80_6	1.03	0.96	$\prec$	1.15	3.18	$\prec$
P80_7	1.59	− 0.16	$\sim$	0.57	5.91	$\prec$
P80_8	0.19	− 2.91	$\sim$	2.77	1.27	$\prec$
P80_9	1.47	1.80	$\prec$	1.30	19.17	$\prec$
P80_10	1.18	− 19.64	$\sim$	1.44	9.13	$\prec$
P100_1	0.23	− 0.17	$\sim$	1.01	6.69	$\prec$
P100_2	− 0.24	− 2.81	$\succ$	1.74	0.66	$\prec$
P100_3	0.26	61.30	$\prec$	1.01	5.61(*)	$\prec$
P100_4	0.27	1.04	$\prec$	0.72	6.05	$\prec$
P100_5	− 0.25	1.18	$\sim$	1.15	1.06	$\prec$
P100_6	− 0.63	50.88	$\sim$	4.77	8.85(*)	$\prec$
P100_7	0.00	− 3.17	$\sim$	3.38	7.08(*)	$\prec$
P100_8	0.45	− 3.41	$\sim$	0.86	4.58	$\prec$
P100_9	− 0.43	86.69	$\sim$	1.44	0.69	$\prec$
P100_10	1.06	− 3.96	$\sim$	27.59	9.29(*)	$\prec$
Pc122-1	− 1.66	− 2.90	$\succ$	4.78	7.44	$\prec$
Pc122-2	0.80	36.42	$\prec$	1.01	6.84	$\prec$
Pc122-3	0.81	4.21	$\prec$	0.57	7.53	$\prec$
Pc224-1	0.18	− 1.63	$\sim$	0.72	2.29	$\prec$
Pc224-2	0.34	44.39	$\prec$	1.29	1.51	$\prec$
Pc224-3	1.26	62.10	$\prec$	0.57	6.19	$\prec$
Pc226-1	− 0.41	51.65	$\sim$	1.00	2.13	$\prec$
Pc226-2	1.40	− 3.06	$\sim$	2.78	− 0.67	$\sim$
Pc226-3	− 0.19	− 0.61	$\succ$	2.47	1.94	$\prec$
Pc238-1	0.00	− 0.50	$\sim$	0.86	− 1.61	$\sim$
Pc238-2	− 0.57	− 11.83	$\succ$	0.86	0.23	$\prec$
Pc238-3	− 0.77	48.78	$\sim$	1.00	− 2.05	$\sim$

One should also add that the results in Table 5 compared with the results of Table 4 reveal that a single run of the algorithms is not sufficient to produce an adequate approximation of the true front. As a matter of fact, the average results per run (Table 4) elect WASP as the best algorithm, while the analysis of the final approximations of the Pareto front produced by the different algorithms (Table 5) clearly suggests the election of MASP.

In Table 6, we evaluate the potentially lexicographic individuals obtained with MASP—lex_1^{MASP} and lex_2^{MASP}—against the best approximation of the lexicographic solutions obtained from a mathematical programming approach—lex_1^{solver} and lex_2^{solver}. This last methodology is based on a bi-objective mixed binary linear programming (MILP) model which is a linearization of the

formulation presented in the Appendix. The MILP model has extra constraints required to transform the two objective functions, f_1 and f_2, into linear functions. The two optimizations needed to attain each of the lexicographic solutions were performed by running the MILP solver of CPLEX 12.2 (ILOG, 2010) with a limit of four hours of CPU time.

Column 2 in Table 6 presents the relative difference (in percentage) between the value of f_1 in lex_1^{MASP} and the corresponding value in lex_1^{solver}, $\Delta f_1 = \frac{f_1^{MASP} - f_1^{solver}}{f_1^{solver}} 100$. Column 3 presents the counterpart relative difference (in percentage) for the value of f_2 again within lex_1^{MASP} and lex_1^{solver}. Column 4 displays the dominance relation between the above lexicographic approximations, obtained from MASP, and the one obtained through the solver, where: $\succ$ means that $lex_1^{MASP} \succ lex_1^{solver}$ (lex_1^{MASP} dominates lex_1^{solver}); $\sim$ means that there is no dominance relation between lex_1^{MASP} and lex_1^{solver}; and $\prec$ means that $lex_1^{solver} \succ lex_1^{MASP}$. Columns 5–7 show similar information respecting lex_2^{MASP} and lex_2^{solver}. In Table 6 results labeled with (*) indicate that optimum objective function values were reached by the solver in the respective optimization. The results show that for 22 in 32 cases the lex_1 approximated by MASP dominates or is equivalent to the one reached by the MILP solver of CPLEX. Only for a single instance, P80_5, the MILP solver was able to attain the true lexicographic solution within the time limit, as it reached the optimum values in both optimizations. Concerning lex_2 MILP clearly outperformed MASP, as MASP was only able to reach solutions equivalent to those of MILP for three instances.

Regarding the computational effort, for each instance, MASP spent on average 5.3 minutes to approximate the lexicographic solutions (Table 4, column 17) while CPLEX ran for four hours which were completely consumed in 58 out of the 64 optimizations (not displayed in the table). In addition, each run of MASP produces a Pareto approximated front comprehending many solutions within a small computational effort of 8.8 minutes on average (Table 4, columns 17–18).

6. Conclusion

This paper is devoted to a new memetic evolutionary heuristic for a Bus Driver Rostering Problem. This specially devised algorithm is compared with two simplified versions of the same evolutionary algorithm, but without local search, using three sets of instances obtained from real and randomly generated data, and taking into account the Law and institutional rules for a Portuguese urban bus company. It is worth emphasizing that the BRP is highly constrained, including six groups of time constraints all considered hard. The innovation of the proposed memetic algorithm consists of introducing in the mating pool of each generation two utopian and two potentially lexicographic individuals, all previously computed by a single objective evolutionary algorithm, an enhanced decoder heuristic, and a specially devised local search procedure.

The computational results revealed that the adopted strategy, though conceptually simple and efficient, is quite effective at improving the approximation of the Pareto front.

The proposed approach is adequate to real-life applications as it can provide planners with a wide variety of potentially efficient rosters that are difficult to obtain manually, thus offering enriched information to support decision-making regarding the most preferable solution(s).

Moreover, this memetic algorithm may be easily adapted to other hard rostering problems. In fact, as the constraints are considered as rules to be imposed through the constructive generator of rosters, additional constraints can be easily introduced.

Acknowledgements

This work was partially supported by Portuguese National Funding from the FCT – Fundação para a Ciência e a Tecnologia, under the project: PEst-OE/MAT/UI0152.

References

Alward, R.R., Monk, T.H., 1993. *The Nurse's Shift Work Handbook*. American Nurses Publishing, Washington, D.C.

Bai, R., Burke, E.K., Kendall, G., Li, J., Mccollum, B., 2010. A hybrid evolutionary approach to the nurse rostering problem. *IEEE Transactions on Evolutionary Computation* 14, 580–590.

Beddoe, G., Petrovic, S., Li, J., 2009. A hybrid metaheuristic case-based reasoning system for nurse rostering. *Journal of Scheduling* 12, 99–119.

Bianco, L., Bielli, M., Mingozzi, A., Ricciardelli, S., Spadoni, M., 1992. A heuristic procedure for the crew rostering problem. *European Journal of Operational Research* 58, 272–283.

Burke, E., Landa Silva, J., 2005. The design of memetic algorithms for scheduling and timetabling problems. In Hart, W., Smith, J. and Krasnogor, N. (eds) *Recent Advances in Memetic Algorithms*, Studies in Fuzziness and Soft Computing. Springer-Verlag, Berlin/Heidelberg, pp. 289–311.

Burke, E.K., De Causmaecker, P., De Maere, G., Mulder, J., Paelinck, M., Vanden Berghe, G., 2010. A multi-objective approach for robust airline scheduling. *Computers & Operations Research* 37, 822–832.

Cappanera, P., Gallo, G., 2004. A multicommodity flow approach to the crew rostering problem. *Operations Research* 52, 583–596.

Caprara, A., Toth, P., Vigo, D., Fishetti, M., 1998. Modeling and solving the crew rostering problem. *Operations Research* 46, 820–830.

Carraresi, P., Gallo, G., 1984. A multi-level bottleneck assignment approach to the bus drivers' rostering problem. *European Journal of Operational Research* 16, 163–173.

Catanas, F., Paixão, J., 1995. A new approach for the crew rostering problem. In Daduna, J., Branco, I. and Paixão, J. (eds) *Computer Aided Transit Scheduling, Lecture Notes in Economics Mathematical Systems*. Springer-Verlag, Berlin/Heidelberg/New York, pp. 267–277.

Chen, J., Lin, Q., Ji, Z., 2010. A hybrid immune multiobjective optimization algorithm. *European Journal of Operational Research* 204, 294–302.

Chu, S.C.K., 2007. Generating, scheduling and rostering of shift crew-duties: Applications at the Hong Kong International Airport. *European Journal of Operational Research* 177, 1764–1778.

Coello, C., Lamont, G.B., Van Veldhuizen, D.A., 2007. *Evolutionary Algorithms for Solving Multi-Objective Problems* (2nd edn). Springer Science + Business Media, New York.

Collette, Y., Siarry, P., 2004. *Multiobjective Optimization: Principles and Case Studies* (2nd edn). Springer, Berlin.

De Matta, R., Peters, E., 2009. Developing work schedules for an inter-city transit system with multiple driver types and fleet types. *European Journal of Operational Research* 192, 852–865.

Deb, K., Agrawal, S., Pratap, A., Meyarivan, T., 2000. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation: NSGA-II. In: Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J.J. and Schwefel, H.-P. (eds) *Parallel Problem Solving from Nature-PPSN VI, 6th International Conference*. Springer-Verlag, Berlin/Heidelberg, pp. 849–858.

Dornberger, R., Frey, L., Hanne, T., 2008. Single and multiobjective optimization of the train staff planning problem using genetic algorithms. *Proceedings of the IEEE Congress on Evolutionary Computation*, CEC 2008, June 1–6, 2008, Hong Kong, China, IEEE 2008, pp. 970–977.

Emden-Weinert, T., Kotas, H.-G., Speer, U., 2001. DISSY—a driver rostering system for public transport, version 1.11. DISSY Project of Programme ESPRIT.

Ernst, A.T., Jiang, H., Krishnamoorthy, M., Owens, B., Sier, D., 2004. An annotated bibliography of personnel scheduling and rostering. *Annals of Operations Research* 127, 21–144.

Freling, R., Lentink, R.M., Wagelmans, A.P.M., 2004. A decision support system for crew planning in passenger transportation using a flexible branch-and-price algorithm. *Annals of Operations Research* 127, 203–222.

Garey, M.R., Johnson, D.S., 1979. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, San Francisco, CA.

Goldberg, D. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, Reading MA.

Hartog, A., Huisman, D., Abbink, E.J.W., Kroon, L.G., 2009. Decision support for crew rostering at NS. *Public Transport* 1, 121–133.

Huisman, D., Freling, R., Wagelmans, A., 2005. Multiple-depot integrated vehicle and crew scheduling. *Transportation Science* 39, 491–502.

ILOG 2007. *CPLEX 11.0, User's Manual*. ILOG CPLEX Division, Incline Village, NV, 2007.

ILOG 2010. *CPLEX Optimization Studio V12.2, User's Manual*. IBM ILOG.

Knowles, J., Corne, D., 2005. Memetic algorithms for multiobjective optimization: issues, methods and prospects. In Hart, W., Smith, J. and Krasnogor, N. (eds) *Recent Advances in Memetic Algorithms*, Studies in Fuzziness and Soft Computing. Springer, Berlin/Heidelberg, pp. 313–352.

Landa-Silva, D., Le, K., 2008. A simple evolutionary algorithm with self-adaptation for multi-objective nurse scheduling. In Cotta, C., Sevaux, M. and Sörensen, K. (eds) *Adaptive and Multilevel Metaheuristics*, Studies in Computational Intelligence. Springer, Berlin/Heidelberg, pp. 133–155.

Lezaun, M., Pérez, G., Maza, E.S.D.L., 2006. Crew rostering problem in a public transport company. *Journal of the Operational Research Society* 57, 1173–1179.

Lučić, P., Teodorović, D., 2007. Metaheuristics approach to the aircrew rostering problem. *Annals of Operations Research* 155, 311–338.

Maenhout, B., Vanhoucke, M., 2010. A hybrid scatter search heuristic for personalized crew rostering in the airline industry. *European Journal of Operational Research* 206, 155–167.

Mesquita, M., Moz, M., Paias, A., Paixão, J., Pato, M., Respício, A., 2011. A new model for the integrated vehicle-crew-rostering problem and a computational study on rosters. *Journal of Scheduling* 14, 319–334.

Mesquita, M., Paias, A., Respício, A., 2009. Branching approaches for integrated vehicle and crew scheduling. *Public Transport* 1, 21–37.

Michalewicz, Z., 1999. *Genetic Algorithms + Data Structures* (3rd edn). Springer-Verlag, Berlin/Heidelberg/New York.

Montalva, S., Muñoz, J., Paredes, R., 2010. Assignment of work shifts to public transit drivers based on stated preferences. *Public Transport* 2, 199–218.

Moz, M., Pato, M.V., 2007. A genetic algorithm approach to a nurse rerostering problem. *Computers & Operations Research* 34, 667–691.

Moz, M., Respício, A., Pato, M., 2009. Bi-objective evolutionary heuristics for bus driver rostering. *Public Transport* 1, 189–210.

Paech, G.M., Jay, S.M., Lamond, N., Roach, G.D., Ferguson, S.A., 2010. The effects of different roster schedules on sleep in miners. *Applied Ergonomics* 41, 600–606.

Pedrosa, D., Constantino, M., 2001. Days-off scheduling in public transport companies. In Voß, S. and Daduna, J.R. (eds) *Computer-Aided Scheduling of Public Transport*, Lecture Notes in Economics and Mathematical Systems. Springer, Berlin/Heidelberg, pp. 215–232.

Schott, J.R., 1995. *Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization*. M.Sc. thesis, Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, USA.

Sodhi, M.S., Norris, S., 2004. A flexible, fast, and optimal modeling approach applied to crew rostering at London underground. *Annals of Operations Research* 127, 259–281.

Van Veldhuizen, D.A. 1999. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. Ph.D. thesis, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH.

Ybema, J.F., Smulders, P.G.W., Bongers, P.M., 2010. Antecedents and consequences of employee absenteeism: a longitudinal perspective on the role of job satisfaction and burnout. *European Journal of Work and Organizational Psychology* 19, 102–124.

Zhang, Q., Li, H., 2007. MOEA/D: a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation* 11, 712–731.

Zitzler, E., Laumanns, M., Thiele, L., 2002. SPEA2: improving the strength Pareto evolutionary algorithm for multiobjective optimization. In K.C. Giannakoglou et al. (eds), *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems* (EUROGEN 2001), International Center for Numerical Methods in Engineering (CIMNE), pp. 95–100.

Zitzler, E., Thiele, L., 1999. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation* 3, 257–271.

Appendix

To present a mathematical formulation of the BRP, the following parameters and sets are introduced:

$V =$ pool of drivers;
$nd =$ number of drivers in the fixed working pool;
$g =$ maximum number of consecutive days without a day off;
$T_h =$ set of duties of day h, $h = -g+1, \ldots, 0,1, \ldots, 28$ (including days of the previous period);
$T_h^{\vartheta} =$ $\{\vartheta\}$, the set with the day off for each day h, $h = -g+1, \ldots, 0,1, \ldots, 28$;
$T_{ih} =$ set of duties that any driver can be assigned to on day h, given that he performed duty i on day h–1, $h = 1, \ldots, 28$, $i \in T_{h-1}$;
$t_{ih} =$ length of duty i on day h, $i \in T_h$, $h = 1, \ldots, 28$;
$\bar{t} =$ contractual daily working time of a driver;
$t'_{ih} =$ max $\{0, t_{ih} - \bar{t}\}$, $i \in T_h$, $h = 1, \ldots, 28$;
$b_1 =$ maximum total work time per week per driver;
$b_2 =$ maximum total work time per rostering period per driver;
$d_w =$ minimum number of days off per week per driver;
$d_s =$ minimum number of Sundays off per rostering period per driver;
$e^v_{ih} =$ 1, if driver v performed duty i on day h of the previous rostering period; or 0, otherwise, $v \in V$, $i \in T_h$, $h = -g+1, \ldots, 0$;
$c^v =$ cost of the overtime hour of driver v ($v \in V$);
$c^S =$ salary cost per extra driver above nd.

The following binary variables are defined:

$y^v_{ih} =$ 1 – if driver v performs duty i on day h or, in case $i = \vartheta$, gets a day off on that day, 0 otherwise, $v \in V$, $i \in T_h \cup T_h^{\vartheta}$, $h = 1, \ldots, 28$;
$\eta^v =$ 1 – if driver v is assigned to at least one duty during the rostering period, 0 otherwise, $v \in V$.

The formulation follows:

minimize (f_1, f_2), where

$$f_1 = \max_{v \in V} \sum_{h=1}^{28} \sum_{i \in T_h} t_{ih} y^v_{ih} \tag{A1}$$

$$f_2 = \sum_{h=1}^{28} \sum_{v \in V} \sum_{i \in T_h} c^v t'_{ih} y^v_{ih} + c^S \max \left\{ 0, \sum_{v \in V} \eta^v - nd \right\} \tag{A1'}$$

subject to:

$$\sum_{v\in V} y_{ih}^{v} = 1, \quad i \in T_h, \quad h = 1, \ldots, 28 \tag{A2}$$

$$\sum_{i\in T_h \cup T_h^{\vartheta}} y_{ih}^{v} = 1, \quad v \in V, \quad h = 1, \ldots, 28 \tag{A3}$$

$$y_{i,h-1}^{v} + \sum_{j\in T_h \setminus T_{ih}} y_{jh}^{v} \le 1, \quad v \in V, \quad i \in T_{h-1}, \quad h = 2, \ldots, 28 \tag{A4}$$

$$e_{i0}^{v} + \sum_{j\in T_1 \setminus T_{i1}} y_{j1}^{v} \le 1, \quad v \in V, \quad i \in T_0 \tag{A4'}$$

$$\sum_{\ell=0}^{g} \sum_{i\in T_{h+\ell}} y_{i,h+\ell}^{v} \le g, \quad v \in V, \quad h = 1, \ldots, 28-g \tag{A5}$$

$$\sum_{\ell=h}^{0} \sum_{i\in T_\ell} e_{i\ell}^{v} + \sum_{\ell=1}^{h+g} \sum_{i\in T_\ell} y_{i\ell}^{v} \le g, \quad v \in V, \quad h = -g+1, \ldots, 0 \tag{A5'}$$

$$\sum_{h=7(\ell-1)+1}^{7\ell} y_{\vartheta h}^{v} \ge d_w, \quad v \in V, \quad \ell = 1, \ldots, 4 \tag{A6}$$

$$\sum_{\ell=1}^{4} y_{\vartheta,7\ell}^{v} \ge d_{\mathrm{s}}, \quad v \in V \tag{A7}$$

$$\sum_{h=7(\ell-1)+1}^{7\ell} \sum_{i\in T_h} t_{ih}\, y_{ih}^{v} \le b_1, \quad v \in V, \quad \ell = 1, \ldots, 4 \tag{A8}$$

$$\sum_{h=1}^{28} \sum_{i\in T_h} t_{ih}\, y_{ih}^{v} \le b_2, \quad v \in V \tag{A9}$$

$$\sum_{h=1}^{28} \sum_{i\in T_h} y_{ih}^{v} - 28\,\eta^{v} \le 0, \quad v \in V \tag{A10}$$

$$y_{ih}^{v} \in \{0,1\}, \quad v \in V, \quad i \in T_h \cup T_h^{\vartheta}, \quad h = 1, \ldots, 28 \tag{A11}$$

$$\eta^{v} \in \{0,1\}, \quad v \in V. \tag{A12}$$

Equalities (A2) and (A3) impose constraints (h1) and (h2), respectively. Constraints (A4) and (A4′) force constraints (h3) as a result of the definition of compatible duties. Inequalities (A5) and (A5′) formalize constraint (h4), whereas (h5) to (h8) are forced by (A6) to (A9). Inequalities (A10) along with minimization of f_2 impose the value 1 for the variable η^v, when driver v is assigned to any duty during the period, otherwise it is set equal to 0, for each $v \in V$. Finally, conditions (A11) and (A12) define the domain of the variables.