A New Hybrid Heuristic for Driver Scheduling

G. R. Mauri[1,2] and L. A. N. Lorena[2]
[1]Rural Engineering Department - Center for Agrarian Science - CCA
Federal University of Espírito Santo - UFES
Alegre - ES - BRAZIL
[2]Laboratory for Computing and Applied Mathematics - LAC
National Institute for Space Research - INPE
São José dos Campos - SP - BRAZIL
E-mail: mauri@cca.ufes.br, lorena@lac.inpe.br

Abstract

This paper describes a new hybrid method based on the application of the Population Training Algorithm (PTA) and linear programming (LP) for generation of schedules for drivers in a public transport system. These methods are applied in an iterative way, where PTA is responsible for the generation of good columns (low cost and good covering of the tasks), and LP for solving a set partitioning problem formed by these columns. The PTA employs heuristics in fitness definition, guiding the population to settle down in search areas where the individuals cannot be improved by such heuristics. The driver schedules are represented by columns in a large-scale set partitioning problem, which are formed when solving the linear programming relaxation. The computational results are compared against a Simulated Annealing metaheuristic using randomly formed instances based on real problems.

Keywords: Driver Scheduling, Population Training Algorithm, Linear Programming, Column Generation.

1. Introduction

Since the 60's research has been done to the development of models dedicated to driver scheduling generation satisfying imposed labor constraints and providing the smallest possible cost. This is a permanently studied problem, since the transport system realities are in continuous transformation and demand efficient management of the available resources. This increasing interest is confirmed by specialized conferences in the last decades, as for example the *International Conference on Computer-Aided Scheduling of Public Transport* (see [17], [20], [23], [5], [4], [21] and [3]).

In spite of this, little has been done in Brazil, where the number of constraints and syndicates rules becomes the problem more complex than in most of the developed countries.

Besides, the access democratization of the public transport system in Brazil depends on the warranty of tolerable tariffs for the users, which depends so much of efficiency increase and reduction of operational costs, as well as of subsidies concession in specific situations under social control. It is well-known that in urban public transport companies the operational labor represents one of the largest costs, so a small reduction can represent a considerable gain in the total cost, reducing significantly tariffs for the final user, justifying any effort in the sense of minimizing the costs with labor.

Several approaches appeared before in literature to solve the driver scheduling problem. The problem is known to be NP-hard (difficult for large instances).

Metaheuristics such as Genetic Algorithms, Tabu Search, Simulated Annealing, among others, allow to easily include several types of labor conditions (see [13], [10], [19] and [8]). Column generation techniques also shown very good results (see [16], [22] and [3]).

This work presents a new alternative to form good driver schedules, the evolutionary approach called Population Training Algorithm (PTA), which forms high quality columns to a set partitioning formulation solved by a column generation technique.

The use of evolutionary algorithms to solve sub-problems in column generation was suggested in [12] for graph coloring instances.

The paper is organized as follows. Section 2 presents the problem definition and modeling. The PTA approach is detailed in section 3, while section 4 describes the column generation process and the PTA/Linear Programming iteration. The computational results are presented in section 5, and conclusions are summarized in section 6.

2. Problem description

The Driver Scheduling Problem [6], also known in some European countries as Bus Driver Scheduling Problem - BDSP [13] or Crew Scheduling [19], consists in assigning to the drivers and collectors (crews) the job of driving the vehicles, in such a way that the trips of the different lines assisted by the company are performed with the smallest possible cost.

This process of assigning tasks to the drivers, also called of driver scheduling, is the construction of a group of legal shifts that cover all the schedule blocks of a vehicle, which is part of a large schedule of vehicles reflecting all the operations of an organization [5].

In the public transport, usually the driver scheduling is made after programming the vehicles, where the trips are gathered in *Blocks* (see Fig. 1), and a block presents the sequence of trips distributed successively to a vehicle beginning and finishing in the garage.

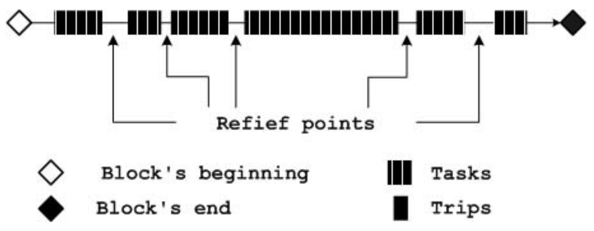

Fig. 1: Vehicle’s block representation.

Each block also shows the *Relief Points* (or *Relief Opportunities* [5]), that are intervals of feasible time to driver changing (see Fig. 1). Through the trips of the vehicles’ blocks, the *Tasks* are formed, and the *Shifts* are created (see Fig. 2), and finally, these shifts are assigned to drivers, which will go to perform them during a working day.

The formation of these shifts (the driver scheduling) must consider a group of rules that are specific to an organization. These rules are usually derived from national and local rules, being obligatory or not. Typically, there are constraints in the total time worked, in the total extension of the shift (duration between the start and the end of the shift) etc.

The modeling approached in this work takes as a base the entities: *Tasks*, *Drivers* (or *Crews*) and *Shifts* (*daily duties*). The problem is described as the construction of a matrix, where drivers appear in columns and tasks in lines. Each element $a_{ij} \in \{0, 1\}$, $i \in M = \{1, ..., m\}$ and $j \in N = \{1, ..., n\}$, where m is the number of tasks (lines), and n the number of drivers (columns), and $a_{ij} = 1$ if the task i belongs to j driver’s shift, and 0 otherwise.

This matrix will be used to solve the following set partitioning problem (SPP):

$$\text{Minimize: } \sum_{j=1}^{n} c_j x_j \qquad (1)$$

$$\text{Subject to: } \sum_{j=1}^{n} a_{ij} x_j = 1 \qquad i = 1, ..., m \qquad (2)$$

$$x_j \in \{0,1\}; \qquad j = 1, ..., n \qquad (3)$$

where: c_j is the cost of column j and x_j is equal to 1 if column j belongs to the solution, and 0 otherwise. This is a classical formulation, constantly used in the literature (see [16] and [22]).

The costs considered in this work penalize extra times, idle times and overlapping times. They are formed by

$$c_j = (ET_j + OT_j) \times w_e + (OV_j + IT_j) \times w_{ne} \qquad (4)$$

where:

$$ET = \max(0, [FT(t_{lst}) - ST(t_{fst})] - MWT)$$

$$OV = \max(0, [FT(t_{lst}) - ST(t_{fst})] - NWT)$$

$$OT = \sum_{i=1}^{m-1} \max(0, [FT(t_i) - ST(t_{i+1})])$$

$$IT = \max(0, NWT - [FT(t_{lst}) - ST(t_{fst})]) + ...$$
$$... \sum_{i=1}^{m-1} \max(0, [ST(t_{i+1}) - FT(t_i)])$$

c_j is the cost of the driver j; ET_j is the piece of time (in total time worked for j driver) that exceeds the maximum working time (*MWT*) (time after 8 hours + 2 extra hours); OT_j is the overlapping time of driver j (time that a driver is, theoretically, accomplishing two tasks at the same time); OV_j is the extra-time (overtimes) in the normal time of work (*NWT*) fixed by law to driver j (8 hours); IT_j is the idle time of driver j (time the driver is not working during his shift); w_e is a penalty weight factor to not attending the essential constraints; w_{ne} is a penalty weight factor to not attending the non-essential constraints; *MWT* is the maximum working time (8 hours + 2 hours); *NWT* is the normal working time (8 hours); $FT(t_i)$ is the finish time of the task i; $ST(t_i)$ is the start time of the task I; t_{fst} is the first task of the j driver's shift; t_{lst} is the last task of the j driver's shift.

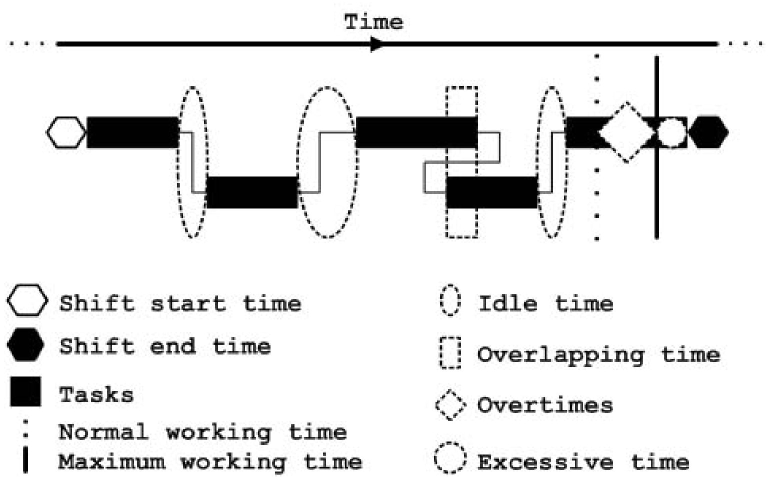

Fig. 2: Shift representation.

To the company is essential (obligatory) to avoid extra times and overlapping but the idle and overtimes are tolerable (non-essential). Columns with cost zero are the best ones, but should not give a good shift. The SPP requires that all the costs be minimized while all the tasks are performed.

The number of possible columns (shifts) of matrix *A* can be huge, and the PTA will be used to generate good columns, reducing the number of columns for the final set partitioning model.

3. Population Training Algorithm

The Population Training Algorithm – PTA (see Fig. 3) is a kind of evolutionary technique first employed in [1], and derived of the Constructive Genetic Algorithm (CGA).

The CGA was proposed by [15] and has a number of innovative features compared to traditional genetic algorithms. These include a ranked population of dynamic size composed of schemata and structures. The schemata and structures are directly evaluated in a common basis, using a double fitness process, called *fg-fitness*.

The schemata are not used in PTA and the fg-fitness will be performed by heuristics. An individual is considered well adapted if it cannot be better regarding the employed training heuristic. The adaptation in the population training is, therefore, used to guide the search to promising areas.

The representation of a shift (inside of PTA) is given by: $S_k = (t_i ,t_j , ..., t_k)$ or, for example, $S_k = (4, 28, ..., 65)$, meaning that the tasks 4, 28, ..., 65 ($i, j, \dots, k$) are in the k driver's shift.

```
Initialize the parameters;
create(a random initial population);
while(gen. number < max. number gen.)
  select(base_column);
  select(guide_column);
  new_column := crossover(base,guide);
  evaluate(new_column);
  if (rand () < mutation percentage)
    mutation(new_column);
    evaluate(new_column);
  end_if
  calculate the rank(new_column);
  if (rank(new_column) > alpha)
    add(new_column in the population);
  end_if
  update(alpha);
  while(exist rank(column) < alpha)
    eliminate(column);
  end_while
end_while
```

Fig. 3: PTA algorithm.

The two functions used in evolutionary training are defined by $g(S_k)$ = cost of shift k (used during the column generation – see expression (7)), and $f(S_k)$ = *Best* $g(S'_k)$ | $S'_k \in$ Neighborhood(S_k) (the training heuristic is detailed in Fig. 4).

The evolutionary process is developed privileging the individuals presenting small differences $[g(S_k) - f(S_k)]$ and small $g(S_k)$ (costs), assigning to them the following ranks:

$$\delta(S_k) = d \times [g_{\max} - g(S_k)] - [g(S_k) - f(S_k)] \qquad (5)$$

where g_{max} is random shift of high cost and d is a constant percentage of g_{max}. The population is dynamically controlled by an evolution parameter, denominated α, and updated as

$$\alpha = \alpha + Step \times PS \times \frac{\delta_{bst} - \delta_{wst}}{RG} \tag{6}$$

where *Step* is a constant that controls the evolutionary process speed, *PS* is the current population size, *(δ_{bst} - δ_{wst})* is the variation among the *ranks* of the best and worst individuals, respectively, and *RG* is an estimated number of remaining generations to finish the process. The parameter α is compared to the ranks in expression (5), and if $\alpha \geq \delta(S_k)$ then the shift S_k is eliminated from the population. The population at the evolution time α is dynamic in size and can be emptied during the process. More details about PTA can be seen in [2].

3.1. PTA operators

For the PTA application in column generation, a simple local search heuristic is employed for the training function *f*, evaluating several (size of the neighborhood) alternative individuals in a neighborhood (see Fig. 4).

```
1.  Let Sₖ be any shift
2.  f(Sₖ) := g(Sₖ);
3.  S'ₖ := clone(Sₖ);
4.  for(x:=1 to size of the neighborhood)
5.    i := select(a task of S'ₖ);
6.    j := select(any task);
7.    remove(the task i of S'ₖ);
8.    add(the task j in S'ₖ);
9.    compute(g(S'ₖ));
10.   if(g(S'ₖ) < f(Sₖ))
11.     f(Sₖ) := g(S'ₖ);
12.   end_if
13. end_for
```

Fig. 4: Training heuristic.

A local search mutation is also implemented selecting a task of a certain driver's shift (randomly) and changing to other, also randomly selected.

This process is repeated while the new shift is invalid, that is, it not assists the essential constraints (see Fig. 5).

The crossover generates valid individuals in the following way: two individuals are selected (base and guide) and merged. The population is kept classified by decreasing values of ranks (as in expression (5)) and the base is selected at the first portion of the population, while the guide is selected from the total population.

To guarantee a valid offspring, a random position is selected in the merged individual, the tasks in previous positions are removed and the subsequent ones inserted in a sequential way, until the new chromosome will be a valid shift (see Fig. 6).

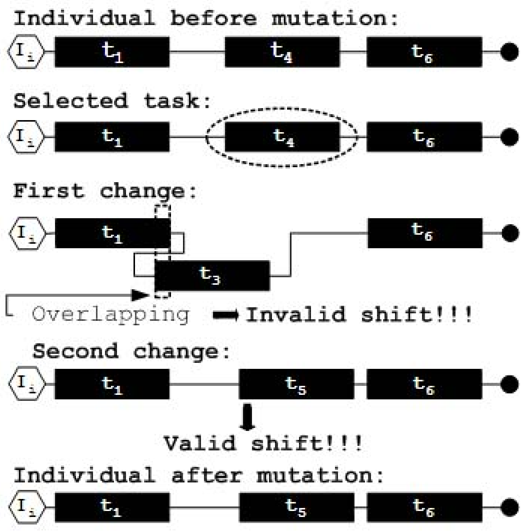

Fig. 5: Mutation.

It is interesting to note that the PTA forms populations of several sizes, guided by the objective of choosing good costs columns (shifts), with the enough covering of the tasks. The best columns will include a diversified number of tasks, characterized by the training heuristic that is driving the evolutionary process.

4. PTA/Linear Programming iteration

Solving the SPP ((1)-(3)) can be a challenge, and the Linear Programming (LP) relaxation is used in conjunction with the PTA to generate a manageable set of columns to commercial solvers [14].

An initial population is randomly generated containing only valid columns (columns that assist the essential constraints) and covering all the tasks.

The LP is solved and new columns are formed through PTA, considering the values of the LP dual variables to construct the fitness functions *f* and *g*. The *g(S$_k$)* function is defined by

$$g(S_k) = \begin{cases} \dfrac{c(S_k)}{\sum_{i=1}^{m} \lambda_i a_{ik}}, for \sum_{i=1}^{m} \lambda_i a_{ik} > 0 \\ c(S_k), for \sum_{i=1}^{m} \lambda_i a_{ik} \leq 0 \end{cases} \tag{7}$$

\

And the corresponding reduced cost:

$$\theta_k = c(S_k) - \sum_{i=1}^{m} \lambda_i a_{ik} \qquad (8)$$

$c(S_k)$ is done by expression (4); θ_k is the reduced cost of column k, and λ_i is the dual variable for task i.

We can observe through expressions (7) and (8) that, for negative reduced costs, the g function values are placed in the interval [0,1]. Therefore, the training heuristic that defines the corresponding function f values (best g in a neighborhood) will assign small differences $(g - f)$ for columns that have negative reduced costs. The population is then indirectly trained for individuals with negative reduced costs (improving columns to LP), preventing the generation of an excessive number of columns, and consequently, speeding up the process.

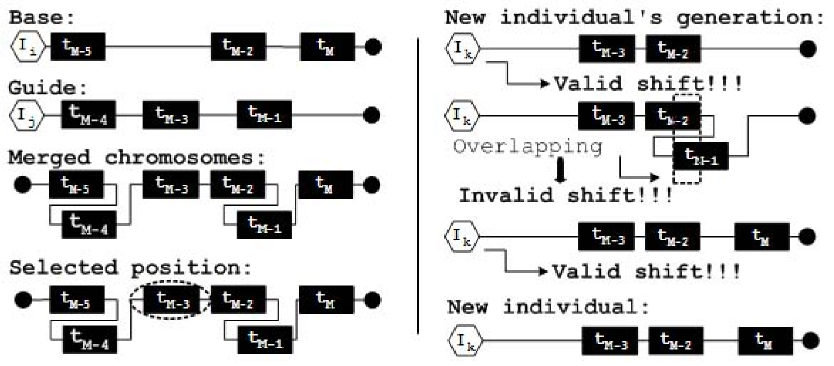

Fig. 6: Crossover.

The columns with negative reduced costs ($\theta_k < 0$) are added to the current LP, and the new SPP is solved again through LP. The process is repeated for a certain number of iterations, and when the solution value of LP stabilizes, or a certain number of iterations be reached, the process is interrupted.

```
1.  create(population addressed to the
           problem);
2.  solve(LP);
3.  while(improving LP)
4.    execute(PTA);
5.    remove(invalid columns);
6.    calculate(reduced costs for new
                columns);
7.    add(columns with negative reduced
           costs);
8.    solve(LP);
9.  end_while
10. convert(LP to ILP);
11. solve(ILP);
```

Fig. 7: PTA/PL algorithm.

Finally the LP is converted to the ILP (Integer Linear Programming), forming the set partitioning problem (expressions (1)-(3)), and solved by the CPLEX [14]. The final problem still could be large to be solved exactly, and so the best feasible solution is kept after some time limit or gap limit (percentage difference of best ILP and LP values).

The algorithm described in Fig. 7 resumes the PTA/LP iteration.

5. Computational results

For validation of the implemented method, some tests were performed on randomly formed instances based on real problems of a Brazilian company of public transport. The number of tasks on instances is: 25, 50, 100, 250 and 500. These instances contain tasks (start time and finish time) randomly selected of a real problem. They are available at *www.lac.inpe.br/~lorena/instancias.html*.

All the tests are performed in a laptop with Intel Celeron® processor of 2.0 GHz and 256Mb of RAM memory. The whole implementation was developed in the C++ language with calls out to the CPLEX library [14].

The parameters' values used by PTA are shown in Table 1, and the computational results are shown in Table 2. Besides, in all the tests, the g_{max} values used were got starting from the largest g function of the generated individuals in the initial population. The value of d was fixed to $1/g_{max}$, in all the tests.

In Table 2, it is also had the results presented by the method PTA/LP*, that was presented in [10]. Basically, the difference among PTA/LP* and PTA/LP is that the first uses the same "weight" to punish the essential and non-essential constraints (with value 1), and PTA/LP uses differentiated weights (with value 1 for the non-essential constraints and with value 1000 for the essentials ones).

Number of tasks	25	50	100	250	500
Size of the initial population	100	100	100	100	100
Step	10^{-17}	10^{-17}	10^{-17}	10^{-17}	10^{-17}
Maximum number of generations	500	1000	1000	1000	500
Base percentage	40	40	40	40	40
Mutation's probability	0.2	0.2	0.2	0.2	0.2
Neighborhood size	7	7	7	7	7

Table 1: PTA parameters.

Number of tasks	Used method	Number of drivers	Overtimes (minutes)	Idle time (minutes)	Total cost	Times (seconds)
25	PTA/LP	12	15	2356	2371	0.10
	PTA/LP*	12	15	2356	2371	0.30
	SA	12	15	2356	2371	1.90
	SA_20	12	15	2356	2371	35.48
50	PTA/LP	20	0	2600	2600	4.63
	PTA/LP*	20	0	2600	2600	41.64
	SA	20	27	2627	2654	42.87
	SA_20	20	0	2600	2600	949.04
100	PTA/LP	40	0	7395	7395	1.88
	PTA/LP*	40	0	7395	7395	4.98
	SA	40	0	7395	7395	7.60
	SA_20	40	0	7395	7395	173.26
250	PTA/LP	81	1103	7751	8854	70.90
	PTA/LP*	82	1058	8186	9244	229.68
	SA	85	815	9383	10198	199.86
	SA_20	83	1112	8720	9832	3749.02
500	PTA/LP	145	1601	8115	9716	6567.80
	PTA/LP*	151	1567	10961	12528	6717.30
	SA	153	1254	11608	12862	7061.52
	SA_20	153	1096	11450	12546	143565.20

Table 2: Computational results.

Besides, PTA/LP* was performed in a "fewer iterative" way, that is, with a lot of PTA generations and few iterations among PTA and LP. The PTA/LP was performed in a "more iterative" way, which few PTA generations and a large number of iterations among PTA and LP than in PTA/LP*.

The PTA/LP approach was capable to form good quality schedules for all instances in reasonable computational times. All the essential constraints were assisted and the nonessential are minimized. The results are compared against a Simulated Annealing (SA) approach applied to an initial set of columns randomly generated and searching on the neighborhood to take moves on the space of solutions. The penalties in column costs are the same defined in expression (4). Two SA solutions are reported differing by the number of executions (SA_20 performed 20 executions for distinct initial sets of columns).

The SA approach used in this work is based on a previous work that got excellent results (best than other methods [9]) for real problems [11].

It is noticed by the Table 2, that for the instances with 25, 50 and 100 tasks PTA/LP got solutions with similar costs to SA_20. The SA got similar solutions to PTA/LP and to SA_20 only for instances with 25 and 100 tasks.

Number of tasks		Test 1	Test 2	Test 3	Test 4	Test 5	Mean time (s)	Deviation (%)
25	Cost	2371	2371	2371	2371	2371	0.10	0.000
	Time	0.10	0.10	0.10	0.10	0.11		
50	Cost	2600	2600	2600	2602	2600	4.73	0.015
	Time	4.63	4.84	4.60	4.74	4.83		
100	Cost	7395	7395	7395	7395	7395	1.57	0.000
	Time	1.88	1.48	1.53	1.49	1.47		
250	Cost	8854	8854	8854	8860	8854	71.34	0.014
	Time	70.90	61.44	98.28	66.58	59.48		
500	Cost	9716	9716	10016	9726	9916	6596.66	1.049
	Time	6567.80	6570.20	6599.76	6637.38	6608.16		

Table 3: PTA/LP - Mean times and deviations.

Besides, in all these cases, the PTA/LP processing times were significantly smaller than both SA based applications and PTA/LP*. For large instances with 250 and 500 tasks, the PTA/LP also obtained better solutions and processing times than SA, SA_20 and PTA/LP*.

Trying to give a measure to the PTA/LP performance, four additional experiments were performed for each instance. These experiments were done starting from the use of random "seeds". They are joined to the experiment reported in Table 2 and used to calculate the deviations (see expression (9)).

$$Deviation = \frac{Cost\_Average - Best\_Cost}{Best\_Cost} \qquad (9)$$

Cost_Average is the average of solutions costs obtained in the experiments and Best_Cost is the smallest (best) known cost for the problems. The deviation and the mean processing time for each instance used in this work are presented in Table 3.

PTA/LP presented a small deviation for all of the instances, and the processing time was practically constant among the experiments, showing a good performance of the proposed method.

Number of tasks	25	50	100	250	500
Number of iterations	3	50	15	700	10000
Number of columns	351	1113	2375	29834	78082

Table 4: PTA/LP details.

Table 4 shows the number of iterations and number of columns generated by the PTA/LP algorithm in computational tests. It should be noted that in the last experiment (500 tasks) described in Table 2, the ILP was interrupted after 6000 seconds of execution.

6. Conclusion

This paper described an application of the PTA to generate good schedules to drives in public transport. The PTA is integrated to traditional column generation techniques and was capable to solve the sub-problem that generates new columns in an implicitly way. This is performed defining properly the fg-fitness with dual prices information. It is somewhat general and can be used in other problems that column generation is an indicated solution method.

PTA was capable to generate columns with negative reduced cost during all the iteration with LP, and the value of the function objective of LP doesn't stabilize, resulting in an excellent solution to the driver scheduling problem. Regarding the method started in [10], the PTA/LP presented in this work improves the iterativity among the PTA and LP techniques. The PTA generates few columns for iteration, but such columns get to improve the LP solution.

The PTA/LP computational results are very good and obtained in reasonable execution times, compared against a Simulated Annealing metaheuristic.

This research can be complemented applying the PTA/LP to driver scheduling instances of the literature and to other NP-hard problems, testing new crossover types (see [7]) and comparing against other hybrid methods, as for example Genetic Algorithms with Linear Programming (see [18]).

7. Acknowledgements

The authors acknowledge FAPESP (process 03/04547-2) and CNPq (process 304598/2003-8) for the financial support given to the development of this work.

References

[1] A. C. M. Oliveira and L. A. N. Lorena, 2-opt Population Training for Minimization of Open Stack Problem, In: Bittencourt, G. and Ramalho, G. L. (eds) Springer Lecture Notes in Artificial Intelligence, Vol. no. 2507, pp. 313–323, 2002.

[2] A. C. M. Oliveira and L. A. N. Lorena, Population Training Heuristics. In EvoCOP 2005, Gottlieb, J. and Raidl, G. R. (eds). Springer Lecture Notes in Computer Science Series, Vol. no. 3448, pp. 166–176, 2005.

[3] A. Dallaire, C. Fleurent and J. M. Rousseau, Dynamic Constraint Generation in CrewOpt, a Column Generation Approach for Transit Crew Scheduling, 9th International Conference on Computer-Aided Scheduling of Public Transport (CASPT), San Diego – California, 2004.

[4] A. Kwan, M. Parker, R. Kwan, S. Fores, L. Proll and A. Wren, Recent Advances in TRACS, 9th International Conference on Computer-Aided Scheduling of Public Transport (CASPT), San Diego – California, 2004.

[5] A. Wren, Scheduling Vehicles and Their Drivers - Forty Years' Experience, 9th International Conference on Computer-Aided Scheduling of Public Transport (CASPT), San Diego – California, 2004.

[6] A. Wren, S. Fores, A. Kwan, R. Kwan, M. Parker and L. Proll, A flexible system for scheduling drivers, Journal of Scheduling, Vol. no. 6, pp. 437–455, 2003.

[7] C. C. Aggarwal, J. R. Orlin and R. P. Tai, Optimized Crossover for the Independent Set Problem, Operations Research 45(2):226-234, 1997.

[8] C. K. Lee, The Integrated Scheduling and Rostering Problem of Train Driver Using Genetic Algorithm, 9th International Conference on Computer-Aided Scheduling of Public Transport (CASPT), San Diego – California, 2004.

[9] E. H. Marinho and M. J. F. Souza, Resolução do Problema de Programação de Tripulações de um Sistema de Transporte Público via Método de Pesquisa em Vizinhança Variável. Technical Report 01/2003, Computer Science Department – Federal University of Ouro Preto, Brazil. 2003. Available on: <http://www.decom.ufop.br/prof/marcone/Orientacoes/PPTviaVNS.pdf>.

[10] G. R. Mauri and L. A. N. Lorena, Driver Scheduling Generation Using a Population Training Algorithm, Proceedings of I Brazilian Workshop On Evolutionary Computation, In: SBRN'04 - Brazilian Symposium in Neural Networks, São Luís, Maranhão – Brazil, 2004.

[11] G. R. Mauri and M. J. F. Souza, Resolução do problema de programação de tripulações de um sistema de transporte público via Simulated Annealing. Technical Report 02/2003, Computer Science Department – Federal University of Ouro Preto, Brazil. 2003. Available on: <http://www.decom.ufop.br/prof/marcone/Orientacoes/PPTviaSimulatedAnnealing.pdf>.

[12] G. Ribeiro Filho and L. A. N. Lorena, Constructive Genetic Algorithm and Column Generation: An Application to Graph Coloring, In Proceedings of APORS 2000 - The Fifth Conference of the Association of Asian-Pacific Operations Research Societies within IFORS, 2000.

[13] H. R. Lourenço, J. P. Paixão and R. Portugal, Multiobjective metaheuristics for the bus-driver scheduling problem, Transportation Science, Vol. no. 35, pp. 331–343, 2001.

[14] ILOG CPLEX 7.5 Reference Manual. September 2001. © Copyright 2001 by ILOG.

[15] L. A. N. Lorena and J. C. Furtado, Constructive genetic algorithm for clustering problems, Evolutionary Computation 9 (3): 309-327, 2001.

[16] M. Desrochers and F. Soumis, A Column Generation Approach to the Urban Transit Crew Scheduling Problem, Transportation Science 23: 1-13, 1989.

[17] N. H. M. Wilson, Computer-Aided Transit Scheduling, Proceedings of the Seventh International Workshop on Computer-Aided Scheduling of Public Transport, Berlin: Springer-Verlag, 1999.

[18]. R. Clement and A. Wren. Greedy Genetic Algorithms, Optimizing Mutations and Bus Driver Scheduling, 6th. International Workshop on Computer Aided Scheduling of Public Transportation, Lisboa, 1993.

[19] S. Groot and D. Huisman, Vehicle and Crew Scheduling: Solving Large Real-World Instances with an Integrated Approach, 9th International Conference on Computer-Aided Scheduling of Public Transport (CASPT), San Diego – California, 2004.

[20] S. Voβ and J. Daduna, Computer-Aided Scheduling of Public Transport, Lecture Notes in Economics and Mathematical Systems, 505, Springer, Berlin, pp. 73-90, 2001.

[21] V. Gintner, N. Kliewer and L. Suhl, A Crew Scheduling Approach for Public Transit Enhanced with Aspects from Vehicle Scheduling, 9th International Conference on Computer-Aided Scheduling of Public Transport (CASPT), San Diego – California, 2004.

[22] W. E. Wilhelm, "A Technical Review of Column Generation in Integer Programming", Optimization and Engineering, 2, pp. 159-200, 2001.

[23] Y. Shen and R. S. K. Kwan, "Tabu Search for driver scheduling", in S. Voβ and J. R. Daduna (eds), Computer-Aided Scheduling of Public Transport, Springer-Velag, Berlin, pp. 121-135, 2001.