7e Conférence Internationale de MOdélisation et SIMulation - MOSIM'08 - du 31 mars au 2 avril 2008 – Paris- France
« Modélisation, Optimisation et Simulation des Systèmes : Communication, Coopération et Coordination.».

IMPLEMENTATION OF A COLUMN GENERATION HEURISTIC FOR VEHICLE SCHEDULING IN A MEDIUM-SIZED BUS COMPANY

Lucas Maia Alves de Lima

Federal University of Minas Gerais (UFMG)
6627, Pres. Antônio Carlos Avenue
31270-010, Belo Horizonte, MG, Brazil
lucasmaia.bra@gmail.com

Laure Thomä-Cosyns

ESIEE Amiens
14, Quai de la Somme
Amiens, France
thoma@esiee-amiens.fr

ABSTRACT: *The paper reports on the current status of a project concerned with the implementation of a Column Generation Heuristic in co-operation with a French medium-sized bus company. We discuss the modeling approach to the practical problem, as well as the mathematical background that supports the strategy adopted. Finally some preliminary results for artificially-generated instances are presented and some key actions to be taken in the near future are highlighted*

INDEX TERMS: *Bus Scheduling, Multiple-Depots Vehicle Scheduling, Column Generation Heuristic.*

1. INTRODUCTION

In this working paper, we report on the results obtained during the first stage of a European Project concerned with the implementation of a computational tool for solving the multiple-depot vehicle scheduling problem in a French medium-sized bus company.

The Multiple-Depot Vehicle Scheduling (hereafter referred to as MDVSP) is a classical problem in the field of Transportation Science. It consists of determining a minimum-cost covering structure for a set of predefined trips using vehicles provided by several depots. Let n be the cardinality of the set of trips $T = \{T_1, T_2, ..., T_n\}$, each one characterized by a starting and end time (s_i and e_i, respectively, for all $i = 1, \ldots, n$), and let m be the cardinality of the set of depots $K = \{D_1, D_2, ..., D_m\}$ each one characterized by a number of available vehicles (v_k, for all $k = 1, \ldots, m$). Now consider any two trips T_i and T_j, and let τ_{ij} be the travel time between the end point of T_i and the starting point of T_j. We say that the pair (i,j) is compatible if $e_i + \tau_{ij} + \kappa_{ij} \leq s_j$, where κ_{ij} is some parameter defined *a priori*, possibly zero. Moreover, throughout the paper A denotes the set of compatible pairs of trips (ie. arcs, in a graph representation) and c_{ij}^k denotes the cost of covering T_j just after T_i using a vehicle provided by depot $k = 1, \ldots, m$.

The MDVSP is proved to be NP-hard in Bertosi, Carraresi and Gallo (1987, cited in Ribeiro and Soumis, 1994). It has been the focus of academic researches for years (see Ribeiro and Soumis, 1994 for a brief literature review) and the evolution of the approaches developed, as well as the improvements in time and quality of solution, is evident. However, to our knowledge, the transposition of this know-how into the context of transportation companies has not yet been well explored. Our experience has shown that the general feeling among managers, engineers and computer scientists working on these companies is that the models are too complicated and time-demanding to be used in real-life situations. According to our partners, many of the transportation companies, both in private and public sectors, still do not formally consider aspects of optimization theory in their vehicle scheduling planning processes.

In this sense, the goals of this paper are to provide a general overview of the modeling and algorithmical approach used to tackle the problem in co-operation with the company, and to present computational results obtained so far, as well as perspectives for the continuation in the near future.

As a contribution to the state-of-the-art, we present a comparison between two approaches for solving the pricing problem in the scope of the Column Generation Heuristic. Our results strengthen the general guidelines given by Barnhart *et al.* (1998), where it was highlighted the absence of computational experience about this specific issue.

We firstly present in Section II two formulations for the MDVSP model as well as our approach to adapt them to the real context. Section III highlights some important points concerning the theoretical background of the field. Results for randomly-generated instances up to 600 trips and 10 depots are presented and analyzed in section IV. Finally conclusions are drawn in section V concerning the know-how generated and the contributions both to theory and practice.

2. THE MODEL

This section is dedicated first, to the presentation of two mathematical formulations of the MDVSP and

second, to explain the way these models were used to represent the real problem. Hereafter, let $o(k)$ and $\bar{o}(k)$ be the nodes corresponding to the starting and end points at depot $k \in K$, respectively. Moreover, let $G^k = (T \bigcup \{o(k), \bar{o}(k)\}, A^k)$ be the graph defined by all the trips in *T* plus the depot $k \in K$, as well as the existing arcs.

2.1. Multi-Commodity Flow Formulation

We start by introducing the multi-commodity flow formulation of the MDVSP, which has been extensively used in the literature (Ribeiro and Soumis, 1994; Pepin, Desaulniers, Hertz and Huisman, 2006). The decision variable is x_{ij}^k, indicating whether the arc (i, j) will be covered by a vehicle from the k^{th} depot, $k \in K$. The maximum capacity of each depot is given by v_k and, as we had defined previously, c_{ij}^k denotes the cost of covering T_j just after T_i using a vehicle provided by depot *k*. For a more complete description of this formulation and its underlying theory, we address to Ahuja, Magnanti and Orlin (1993).

$$\min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^k x_{ij}^k \qquad (1)$$

$$st \begin{cases} \sum_{k \in K} \sum_{(i,j) \in A^k} x_{ij}^k = 1 & \forall i \in T \quad (2) \\ \sum_{j:(j,i) \in A^k} x_{ji}^k = \sum_{j:(i,j) \in A^k} x_{ij}^k & \forall i \in T; k \in K \quad (3) \\ \sum_{j:(o(k),j) \in A^k} x_{o(k)j}^k \le v_k & \forall k \in K \quad (4) \\ x_{ij}^k \in \{0,1\} & \forall (i,j) \in A^k; k \in K \quad (5) \end{cases}$$

Model 1. MDVSP Multi-Flow Formulation

In Model 1, the minimization of the total cost is established in (1). Constraints (2) impose that exactly one arc has to arrive (to depart) at (from) each node $i \in T$ of the graph while constraints (3) impose that these two arcs (ie. the one arriving at and the one departing from a given node) have to be covered by a vehicle provided by the same depot $k \in K$. In constraint (4) we have that the number of used vehicles per depot is not greater than the number of available vehicles and, finally, (5) imposes binary variables.

2.2. Set Partitioning Formulation

Differently from the multi-commodity formulation, the set partitioning formulation considers a set Ω^k containing some set of feasible schedules provided by depot $k \in K$. The optimization is therefore performed over $\Omega = \bigcup_{k \in K} \Omega^k$. In the specific case of the MDVSP, these variables are represented by $(n+m) \times 1$ vectors where the first *n* positions correspond to the set of trips to be covered and the last *m* positions to the set of depots from where the vehicles may come. For a given variable, say *p*, the i^{th} component a_{ip} is equal to one if *p* covers *i*, and zero otherwise. As for the last *m* positions, a given component, say *i* (with $n+1 \le i \le n+m$), is equal to one if *p* comes from depot $i-n$, and zero otherwise. Finally, computing the cost c_p of each variable *p* in Ω, and defining θ_p equal to one if schedule *p* is taken in the solution and zero otherwise, we can now formulate the set partitioning model as follows.

$$\min \sum_{k \in K} \sum_{p \in \Omega^k} c_p \theta_p \qquad (6)$$

$$st \begin{cases} \sum_{k \in K} \sum_{p \in \Omega^k} a_{ip} \theta_p = 1 & \forall i \in T \quad (7) \\ \sum_{p \in \Omega^k} \theta_p \le v_k & \forall k \in K \quad (8) \\ \theta_p \in \{0,1\} & p \in \Omega^k; k \in K \quad (9) \end{cases}$$

Model 2. MDVSP Set Partitioning Formulation

We will briefly interpret the constraints of

Model 2 comparing its roles with those of Model 1. First, we see that (6) plays the same role of (1), ie. minimizing the total costs of the chosen schedules. Besides, (7) and (2) have also the same role, assuring that all services will be covering once. Finally, (8) plays the same role of (4) and (9) of (5). One may think that (3) is not being considering, but in fact this constraint must be considered when building the set Ω. It means that all the schedules included is Ω already obey to (3). The way this is done will be clearer in Section III.

Although the comparison above shows that both the multi-commodity and the set partitioning formulations are equivalent models for the MDVSP, one can mathematically proof this fact by applying the Decomposition Principle to Model 1. We bypass this proof and refer to Ahuja, Magnanti and Orlin (1993) for details.

We now address the way these mathematical formulations can be used to model the practical problem handled by the company. Our partner possesses three physical depots, located in the

surroundings of Amiens, France and performs two types of trips: urban and inter-urban. The focus of the current project is on the latter, typically longer trips. Besides, the particular characteristic of this application that makes it slightly different from the usual is the fact that the driver may accompany the bus for several days without going back to any of the three physical depots. Instead the bus is taken with the driver to his residence. In this case, considering in the model only the three physical depots would be obviously of no value from the practical point of view, however, on the other extreme, considering each driver's residence as a potential depot with unitary capacity would be impractical from the computational point of view.

Finally, the solution proposed was to divide the geographical region over where the trips are distributed – La Somme, France – in an appropriate number of smaller regions so as to consider the center of each one as a virtual depot. We present the resulting division in

Figure 1. The red and black dots correspond to drivers' residences, while the blue limits are circles in the space of minutes, each of them with a ray equal to 20 minutes. The definition of this ray takes into consideration the assumption that the drivers allocated to each virtual depot (ie. each circle) would travel an extra amount of about 20 minutes to get home at the end of the day and that this would not impact on the practical value of the scheduling for the company. Having this assumption validated we came up with a reasonable number of depots, which could be more easily dealt by the existing algorithms.

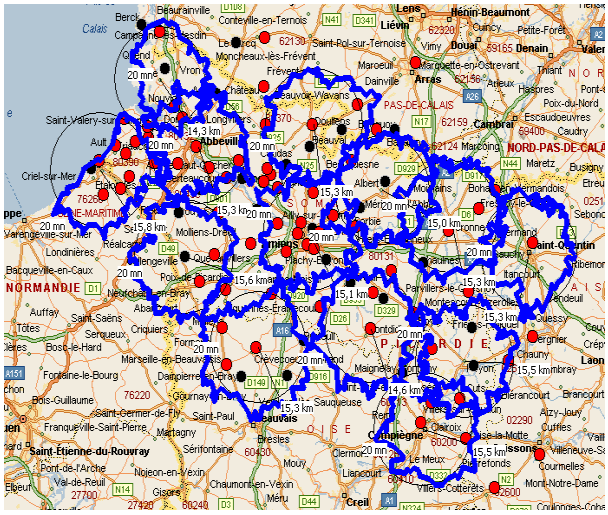

Figure 1. Definition of Virtual Depots

3. BACKGROUND

In this section we present some important theoretical concepts which provide the basis for understanding the Column Generation approach. We explain briefly (3.1.) the roles of the master and pricing problems as well as the relationship between them; (3.2.) the rounding procedure used to rapidly compute integer solutions for the master problem; and (3.3.) the K-shortest path algorithm chosen for solving the pricing problem.

3.1. Column Generation to solve huge linear programs

When dealing with linear programs of very large scale the set of feasible solutions may be so immense that traditional methods, like the Simplex (Chvátal, 1980), may become useless. The reason lies on the fact that an explicit search for priced-out variables to enter the basis is too difficult to be performed efficiently. On the other hand, we observe that the optimal solution of these huge problems usually contains a very tiny part of the feasible set. This observation gives rise to the idea of manipulating just a small subset of the most "interesting" variables among all the possible ones[1].

To tackle the question of how to determine this "interesting" set of variables the Column Generation Algorithm deals with two levels of problems that exchange information throughout the procedure. The first level corresponds to the *Master Problem*, represented in our case by

Model 2. This is the upper level of the algorithm and it manages the set Ω which contains variables from all the depots. The dual information provided by the solution of the Master Problem is then used to modify the arc's costs of each graph $G^k \forall k = 1,...,m$, which will be considered independently in each of the *Pricing Problems*. As the title suggests, the Pricing Problems use the dual information provided by the Master to determine a few priced-out variables (new schedules with negative reduced cost, in our case) that will hopefully improve the current solution, if it is not optimal yet.

One can proof that these Pricing Problems are actually shortest paths problems over each G^k and with an adequate definition of arc's costs, the optimal solution of each shortest path problem also solves to optimality the multi-flow formulation and its equivalent set-partitioning version. Let π_i and β_k be the dual values corresponding to constraints (7) and (8), respectively. Then, choosing $\bar{c}_{o(k),i} = c_{o(k),i} - \beta_k$ $\forall i \in T; k \in K$ and $\bar{c}_{ij} = c_{ij} - \pi_i \; \forall i \in T$, makes

[1] This idea was firstly introduced by Ford and Fulkerson (1958, cited by Lübbecke, 2001) and further developed by Dantzig and Wolfe (1960, cited by Lübbecke, 2001) who pioneered the fundamental idea of decomposition.

the optimality conditions of each shortest path problem equivalent to the ones of the multi-flow problem, validating the resolution of the shortest-path problems independently as a searching engine for priced-out variables. This result is detailed in Lima (2007) based on the proof provided by Ahuja, Magnanti, Orlin (1993).

Having explained the general strategy to deal with the difficulty of the multi-commodity-flow problem and the role of the Master and Pricing Problems, we turn out efforts to understanding how to solve them.

3.2. The Rounding Procedure to solve the Master Problem

In the beginning of the algorithm, the set partitioning formulation given by

Model 2 is initialized with some set of rather expensive artificial variables – in order to assure the existence of a feasible solution to the LP relaxation of the master – which are then progressively eliminated as better variables are added to Ω. The importance of this initial phase lies in the fact that it determines the quality of the dual variables passed to the pricing problem, which thereby determines the quality of the columns generated. In this sense, some important aspects used in our implementation were taken from the discussions in Lübbecke (2001) or are detailed in Lima (2007).

We have chosen to solve the LP relaxation of

Model 2 using the large scale linear programming algorithm described in Zhang (1997). In this sense, one of the main concerns regarding the solution of the master problem is the development of an efficient way to find an IP solution that can be proved to be optimal. Branching strategies with this aim have been extensively studied in the past (see Desaulniers *et al.* 2005 for a survey of these practices), but are not within the scope of this work. As argued by Barnhart *et al.* (1998), if finding a good IP solution is of greater concern over proving optimality – as it is often the case in practical applications – it makes sense to adopt a greedy branching scheme that divides the solution space in non-even sub-trees. This strategy can go even further when backtracking in the tree is not important either, in which case, after solving the LP relaxation, variables with high fractional values are fixed to one. This approach has been used successfully by Marsten (1994, cited by Barnhart, 1998) in the context of crew scheduling and in Pepin, Desaulniers, Hertz and Huisman (2006), in the context of vehicle scheduling. In the following, we provided a formalization of this procedure.

Algorithm 1. Rounding Procedure
Define a rounding threshold γ;
Let $\mathbf{\Pi}$ be the current master problem and $\mathbf{\Gamma}$ be the current corresponding graph;
while *no integer feasible solution was obtained* **do**
  for *all variables* θ_p **do**
    if ($\theta_p \geq \gamma$) **do**
      $\theta_p = 1$;
      Delete the column corresponding to θ_p from $\mathbf{\Pi}$ and all lines corresponding to the trips covered by θ_p;
      Also delete the nodes in $\mathbf{\Gamma}$ corresponding to the trips deleted;
      Update vehicle availability;
    end if
  end for
  if *no variable was rounded* **do**
    Round up to 1 the greatest variable θ_p;
    Delete its corresponding column from $\mathbf{\Pi}$ and all lines
    Corresponding to the trips covered in it;
    Also delete the nodes in $\mathbf{\Gamma}$ corresponding to the trips deleted;
  end if
  Solve the Master Problem over $\mathbf{\Pi}$ and the Pricing Problem over $\mathbf{\Gamma}$;
end while

The definition of the rounding threshold is rather experimental. As suggested by Pepin, Desaulniers, Hertz and Huisman (2006), we have used $\gamma = 0{,}7$ which worked well with our database.

3.3. Using a K-shortest paths algorithm to solve the Pricing Problem

The solution of the pricing problem lies on the very heart of the column generation algorithm. For instance, Lübbecke (2001) reports on exhaustive tests showing that up to 80% of the CPU time may be spent on this phase. For being so expensive, instead of adding just the variable corresponding to the shortest path between $o(k)$ and $\bar{o}(k)$ we would rather add a greater number of variables, all of them potentially capable of improving the current solution. In this sense, we need to determine not only the shortest path in G^k but some others equally "good" paths, which is done by means of a K-shortest path algorithm.

However, as this strategy goes naturally against the will of simplicity evoked in Section I, we need to address the question of whether the choice of a K-shortest path algorithm really provides any gain of performance when compared with a single shortest path approach. This will be demonstrated further, in

Section IV. For the time being, we consider this choice appropriate.

We have decided to use the MPS Algorithm (Martins, Pascoal and Santos, 1998) to solve the K-shortest path problem. According to the authors aforementioned, although the theoretical complexity of this algorithm is still an open question it has performed well compared to its closest competitors and this is the main reason for our choice. The MPS Algorithm requires the construction of a tree of shortest paths, for which we used the polynomial version of the Auction Algorithm with Graph Reduction proposed in Bertsekas, Pallotino and Scutella (1995). This latter was implemented using the "second best" type of implementation and the pre-processing algorithm, both suggested by Bertsekas (1991). Moreover, to improve the performance of the K-Shortest Path algorithm we also used a sorted forward star form to organize the set of candidate paths. Finally, as the sub-problems are solved sequentially, we have chosen to use the information contained in the price vector of one sub-problem to initialize the next, reducing the time spent in the pre-processing phase of the Auction Shortest Path algorithm. The interested reader is addressed to Lima (2007) for details about these performance improvement procedures.

Two interesting effects concerning the resolution of the Pricing Problem are reported in the literature (Lübbecke, 2001; Desaulniers and Desrosiers, 2005). The first one, known as *heading-in*, is related with the possibility of generating an important amount of "bad" columns in the beginning of the algorithm, when lots of artificial variables are still present in the formulation. In this case, the dual information passed to the Pricing Problem is poor and therefore, the columns generated are not good. To avoid this effect, instead of maintaining a constant set point for the K-shortest path algorithm, we rather start with a small desired number of columns to be generated per sub-problem and increase this number as new columns are added and artificial ones are deleted. The second typical effect refers to the fact that improving the total cost becomes more difficult as the algorithm evolves. As a consequence, at the end, the Pricing Problem continues generating columns with negative reduced cost but, the improvement in the solution does not make it worth to continue the process. This happens because proving optimality would be too expensive in terms of time. To avoid this *tailing-off* effect we introduce a parameter that monitors the number of iterations without overall cost improvement and whenever this parameter becomes greater than a given arbitrary threshold (3 to 5, in our experiments), we allow the early termination of the algorithm.

4. RESULTS

Despite the early stage of the project, we have already been able to come up with a first generation of preliminary results for randomly generated data sets. These first rounds of tests aim to evaluate the suitability of the approach developed so far and play an important role as a way to identify gaps where further work is needed.

All codes were implemented on Matlab™. For this reason the running times presented herein are on average greater than the ones achieved by implementations carried out on other languages, especially structured ones. On the other hand, as pointed out by Zhang (1997), Matlab™ allows a much faster development and therefore is suitable for testing the concept of a solution framework before getting deeper in its implementation. All instances were run in an Intel Xeon™ 2,66GHz Workstation, 1 Gb RAM, Windows XP, Matlab 6.5.

The data was generated randomly according to the statistical characteristics of real timetables obtained through the partner's database. A brief description of the parameters is provided in Table 1.

	Variables	Description	Generation
Trips' Characteristics	*startLatitude*	Starting GPS position of the trip	Uniform-ramdomly generated in the square defined by the extreme GPS positions of the area currently covered by the company
	startLongitude		
	endLatitude	End GPS position of the trip	
	endLongitude		
Arcs' Weight	*duration*	Duration of the trip	A linear regression model was built to describe the behavior of the duration as a (linear) function of the start and end positions of a trip. This function is used here to compute the duration of each trip
	travelTime	Travel time between any two compatible trips	Exponential-ramdomly generated with lambda parameter equal to the avearage travel time of a given representative day
	cost	Cost of covering the trip in the head of the arc just after the one in its tail	Linear fonction of travel and idle times

Table 1. Parameters for data generation

We start by investigating whether the K-shortest path algorithm provides any gain in performance that justifies its implementation. For this purpose, the set of graphs shown in Figure 2 compares the solution times of both approaches for instances ranging from 15 to 600 trips, and 2 to 5 depots.

The two graphs on the top half of the picture present this comparison both regarding the elapsed time until the best-cost-solution was found and the total elapsed time of the procedure. As for the former, the K-shortest path approach was faster in 80,77% of the replications and this improvement was greater than 15% in 73,08% of the replications. When it comes to the latter, the K-shortest path approach was faster in 76,93% of the cases, improving the total time by more than 15% in 55,77%.

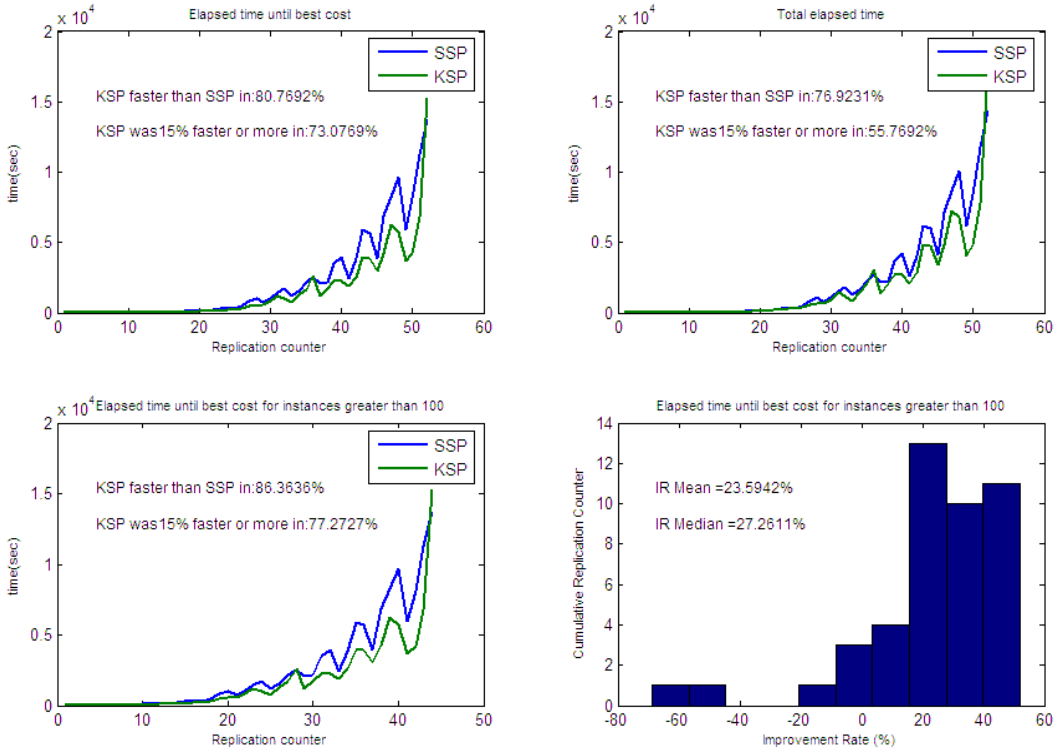

Figure 2: k-Shortest Paths (KSP) vs. Single Shortest Path (SSP)

An interesting insight arrives when analyzing just replications in which the instance had a number of trips greater than 100. In this case, the K-shortest path approach outperforms its competitor in 86,36% of the replicaions, with a contribution of more than 15% in 77.27% of them. For these instances, greater than 100 trips, we present also the distribution of the improvement rates obtained in the histogram on the bottom left of the figure. Therefore, we conclude that the adoption of a K-shortest path algorithm within the scope of the pricing problem provides a fairly significant gain in performance, which justify its use.

These results are coherent with the guidelines given by Barnhart *et al.* (1998) who suggested that adding multiple columns would work better when the pricing problem is computationally intensive. Otherwise, it would be more interesting to add just the column with the highest reduced cost, using therefore a single-shortest path algorithm to solve the Pricing Problem.

Being convinced of the utility of the K-shortest path strategy as well as its potential advantages for a greater number of trips and depots, we can now turn our efforts to analyzing the results obtained for the current version of the code for instances up to 600 trips and 4, 8 and 10 depots. Table 2 shows these figures.

As one should expect the results show a linear dependency between the solution time and the cardinality of the set of depots. This is easily explained by the sequential nature in which the pricing problems are solved and can only be improved if all problems were carried out in parallel. On the other hand, the dependency between the solution time and the cardinality of the set of trips is strongly non-linear, demonstrating the theoretical complexity of the MDVSP.

Trips	Depots	Total Cost (1,0e+006)	Number of Vehicles	Elapsed Time Until Best Solution (1,0e+004)	Total Time (1,0e+004)
50	4	0,1507	13	0,0021	0,0027
	8	0,1503	13	0,0063	0,0095
	10	0,1501	13	0,0101	0,0152
100	4	0,2886	24	0,0024	0,0034
	8	0,3044	25	0,0035	0,0059
	10	0,2653	23	0,0732	0,1085
200	4	0,5106	42	0,0276	0,0360
	8	0,5768	48	0,0300	0,0490
	10	0,5877	49	0,0350	0,0660
300	4	0,7924	64	0,0636	0,0767
	8	0,7191	59	0,1627	0,2205
	10	0,8340	68	0,1523	0,2148
400	4	1,0507	89	0,3741	0,4574
	8	0,8819	73	0,5291	0,7373
	10	1,0085	83	0,5882	0,7869
500	4	1,1466	94	0,6262	0,7139
	8	1,3316	113	1,4039	1,8082
	10	1,2073	100	1,5880	2,0175
600	4	1,3262	108	1,1310	1,3323
	8	1,4604	120	2,0160	2,4894
	10	1,5582	128	2,1879	2,9225

Table 2. Results' compilation using KSP

In the following we finish this section pointing out the most important actions to be taken in the near future in order to improve the results obtained so far.

i. Run of more intensive tests to improve the code's parameters;

ii. Tests with label setting shortest path algorithms: Bertsekas, Pallotino and Scutella (1995) report on results of comparative tests between the version of the Auction Algorithm used here and its label setting competitor. In general the latter performed better although the former is more suitable for parallel computation. We believe that implementing this version and comparing its results to the current ones could be of great value.

iii. Use parallel computation to solve the Pricing Problem and the Auction Algorithm for the Shortest Path Problem: we could use the concept of parallel computation to distribute the solution of each sub-problem of the pricing phase among a given number of processors running in parallel. The result of each of these problems would be then put together and written in a common shared memory. Besides we can go even further implementing the forward-reverse version of the auction algorithm for shortest paths in parallel. In doing so, two processors modifying the same price vector are used; one running the forward and the other the backward auction processes. The execution is stopped when the paths meet at a common node Polymenakos (1991) discusses some parallelization issues specifically designed for the Auction Algorithm.

iv. Implementation in a non-interpreted language: as discussed briefly before, after having a robust version of the current code, the implementation of the codes in some non-interpreted language (C or C++, for example) will provide a quantum leap in the current performance, increasing the suitability of the code for even harder instances.

v. Implementation of a Large Neighborhood Search Heuristic (LNS): as suggested in the experimental work carried out by Pepin, Desaulniers, Hertz and Huisman (2006), the Column Generation Heuristic can be combined with an algorithm for a slightly modified Single Depot Vehicle Scheduling Problem[2] giving rise to a more powerful tool for tackling the MDVSP. According to the results presented by the authors aforementioned the LNS heuristic yields better results in terms of the commitment between quality and solution time.

[2] Despite not being reported in this paper an algorithm for the Single Depot problem was also implemented in the first phase of the work. The work was mostly based on Freeling, Wagelmans and Pinto Paixao (2001).

5. CONCLUSIONS

Throughout this paper we presented the main topics concerning the approach used to tackle a practical scheduling problem in a medium-sized bus company. Despite the early stage of the work some experimental results for real-size randomly generated instances were already presented.

As for the continuation of the project, some key points to be carried out in the near future to ensure the improvement of the current performance were pointed out and explained.

On the one hand, from the theoretical point of view, the paper put together a wide range of techniques from different areas of the Operational Research literature, making it easier for practitioners to get started with similar applications. Also, by comparing the K-shortest path approach with the single one we could demonstrate the gain of performance obtained, mainly when dealing with bigger instances, helping to strengthen the computational experience about this issue and supporting our choice for the former alternative. Lastly, given the relatively newness of the auction algorithm, its incorporation within the scope of project contributes to evaluate its advantages and drawbacks when compared with its traditional label setting competitors.

On the other hand, from the practical point of view, the envisaged algorithm will make it easier for the company to answer rapidly and precisely to public demands, providing it with a scientific tool of planning, mainly when dealing with huge instances of thousand of trips. Last, but not least, the work also shows how the co-operation academy-industry can enrich the application of tools developed by the former, encouraging the latter to invest in their development and application.

6. ACKNOWLEDGEMENT

Lucas would like to express his gratitude to Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES, Brazil), for its financial support during his one-year exchange program in France and to the Federal University of Minas Gerais (UFMG) for awarding him with this opportunity.

Together, the authors acknowledge the CAP staff for being continuously enriching the work since its very beginning. Also we are both grateful to the European

Fond of Development for its support through Interreg IIIA

7. REFERENCES

Ahuja, R.K., Magnanti, T.L., Orlin, J.B. Network Flows : Theory, Algorithms and Applications. Prentice-Hall, Englewood Cliffs, New Jersey. 1993

Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.P., Vance, P.H. Branch-and-Price: Column Generation For Solving Huge Integer Programs. *Operations Research. Vol. 46, Nº 3.* May-June 1998

Bertsekas, D.P. The Auction Algorithm for Shortest Paths. *SIAM Journal on Optimization. Vol 1, pp 425-447.* 1991

Bertsekas, D.P., Pallotino, S., Scutella, M.G. Polynomial Auction Algorithms for Shortest Paths. *Journal of Computational Optimization and Applications. Vol 04.* 1995

Chvátal. Linear Programming. New York: W. H. Freeman and Company. 1980

Desaulniers, G., Desrosiers, J., Solomon, M.M. Column Generation. Gerad 25th Anniversary Series. Springer, 1st edition. 2005

Freeling, R., Wagelmans, A.P.M., Pinto Paixao, J.M. Models and Algorithms for Single-Depot Vehicle Scheduling. *Transportation Science. Vol. 35, N°2, pp.165-180*, May 2001

Lima, L.M.A. *Optimisation Combinatoire: une application à la gestion de flottes de bus*. Technical Report. ESIEE Amiens. 2007

Lübbecke, M. Engine Scheduling by Column Generation. PhD Thesis. 2001

Lübbecke, M., Desrosiers, J. Selected Topics in Column Generation. *Operations Research, Vol. 53, N° 6, pp. 1007-1023*, Nov-Dec 2005

Martins, E.Q.V., Pascoal, M.M.B., Santos, J.L.E. Deviation Algorithms for Ranking Shortest Paths. *International Journal of Foundations of Computer Science.* 1999.

Pepin, A.S., Desaulniers, G., Hertz, A., Huisman, D. Comparison of heuristic approaches for the multiple depot vehicle scheduling problem. *Econometric Institute Report*. Nov 2006. Available at: http://hdl.handle.net/1765/8069

Polymenakos, L. Analysis of Parallel Asychronous Schemes for the Auction Shortest Path Algorithm. *MS Dissertation. MIT, Cambridge, MA.* 1991.

Ribeiro. C.C., Soumis, F. A column generation Approach to the multiple-depot vehicle scheduling problem. *Operations Research, Vol. 42, N°1*, Jan-Fev 1994

Zhang, Y. Solving Large-Scale Linear Programs by Interior Point Methods Under the MATLAB™ Environment. 1997