BTPS
Brazilian Transportation Planning Society

Journal of Transport Literature
Vol. 8, n. 4, pp. 227-251, Oct. 2014
Research Directory

JTL | RELIT
www.journal-of-transport-literature.org
ISSN 2238-1031

A study of different metaheuristics to solve the urban transit crew scheduling problem

[Um estudo de diferentes metaheurísticas para resolver o problema de programação de tripulações do sistema de transporte público]

Gustavo Peixoto Silva*, Allexandre Fortes da Silva Reis

Federal University of Ouro Preto, Brazil

Submitted 24 Jun 2013; received in revised form 29 Nov 2013; accepted 7 Apr 2014

Abstract

This paper explores different local search methods associated with the metaheuristic Iterated Local Search (ILS) to solve the Crew Scheduling Problem (CSP) of a Public Transportation System. The results from ILS were compared to those obtained in a previous work from the same authors that used the Variable Neighborhood Search (VNS). Initially, both metaheuristics were implemented using, as local search, the classical First Improvement Method, performing "guided" reallocation and exchange of crew tasks. The guided reallocation/exchange replaces random components from the classical method by searching the best position to insert the task. Further, the Very Large-scale Neighborhood Search (VLNS) technique was used as a local search procedure in the metaheuristics. This technique has substantially more neighbors than the 2-opt neighborhoods, since it performs a chain exchange of tasks from different crews. Both versions of metaheuristics were applied to a set of real data from a company operating in the city of Belo Horizonte, producing more economical schedules than those adopted by the company. The results are presented and discussed in this work.

Key words: crew scheduling problem, metaheuristics, variable neighborhood search, iterated local search.

Resumo

Este artigo explora diferentes métodos de busca associados à metaheurística Iterated Local Search (ILS) para resolver o Problema de Programação de Tripulações de um Sistema de Transporte Público. Os resultados obtidos com o ILS foram comparados com um trabalho prévio, dos mesmos autores, que utilizou a metaheurísica Variable Neighborhood Search (VNS). Inicialmente ambas as metaheurísticas foram implementadas utilizando como procedimento de busca o método clássico First Improvement, realizando realocação e troca "guiada" das tarefas das tripulações. Esta realocação/troca guiada substitui a componente randômica dos métodos clássicos por uma busca da melhor posição para a inserção das tarefas. Posteriormente, foi utilizada a técnica denominada Very Large-scale Neighborhood Search (VLNS) como procedimento de busca nas respectivas metaheurísticas. Esta técnica produz um número muito maior de vizinhos do que vizinhanças 2-opt, pois ela permite a realocação de tarefas entre uma série de diferentes tripulações. Ambas as versões das metaheurísticas foram aplicadas a um conjunto de dados reais de uma empresa que opera em Belo Horizonte, produzindo programações mais econômicas do que aquelas adotadas pela empresa. Os resultados são apresentados e discutidos neste trabalho.

Palavras-Chave: problema de programação da tripulação, metaheurísticas, variable neighborhood search, iterated local search.

* Email: gustavo@iceb.ufop.br.

Recommended Citation

Silva, G. P. and Reis, A. F. S. (2014) A study of different metaheuristics to solve the urban transit crew scheduling problem. Journal of Transport Literature, vol. 8, n. 4, pp. 227-251.

■ *JTL|RELIT is a fully electronic, peer-reviewed, open access, international journal focused on emerging transport markets and published by BPTS - Brazilian Transport Planning Society. Website www.journal-of-transport-literature.org. ISSN 2238-1031.*

This paper is downloadable at dx.doi.org/10.1590/2238-1031.jtl.v8n4a9

Introduction

Currently, companies in all sectors of our economy are interested in technological innovations as a possibility to improve their performances. However, companies in the Brazilian public transportation system make little use of optimization software to schedule their equipment and workers, that is, vehicles and crews. This normally happens because the use of optimization models requires gathering of precise data, compliance with specified rules and flexibility of the system operation, among other practices still not very disseminated in the sector. On the other hand, in more demanding systems such as air transportation, the use of optimization models to schedule their crews is almost mandatory. Besides enabling the generation of possible schedules, the models can decrease the costs with crews, which in the case of air transportation, are very high. Thus, the study and implementation of effective methods to solve the problem of scheduling the crews in the public transport system, as well as the dissemination of the use of the software, are important tasks not only to reduce the operational costs but also to increase the use of computational support systems by the management of companies which work in this sector.

The task of developing a programming of an event is an activity that gradually becomes more and more difficult as the quantity of items to be programmed and the restrictions of the problem both increase. Likewise, the scheduling of the urban transport crews becomes a problem gradually harder to be solved due to the labor and operational constraints involved. Thus, an efficient scheduling must have as minimum cost as possible, increasing productivity and the satisfaction of employees and users of the system.

In this paper the crew scheduling problem of an urban bus company is modeled and solved through the ILS metaheuristic, exploring two different local search procedures. The most significant cost component of a transportation company is due to crew remuneration. Therefore, defining a crew scheduling with minimal cost leads to a great economy for the company. The crew schedule is built from a previously defined vehicle scheduling. So, once vehicle schedule is previously defined, operational rules and labor laws are input data to the problem. Computational tests were performed with real data from a week of operation and the results are presented in this work.

This work is divided as follows: in the following section the CSP tackled in this paper is presented. In Section 2 the most important works about the problem are presented. In Section 3, the local search procedures are presented, in detail. In Section 4 it is described how metaheuristic ILS was implemented to solve the problem. Section 5 shows the results and, in Section 6, these results are discussed. The last section contains the conclusions of this work.

1. The Crew Scheduling Problem

The Crew Scheduling Problem (CSP) consists in determining the minimum number of crews to cover entirely a vehicle schedule previously made. The solution of this problem involves sequencing drivers' activities in order to generate a set of work duties, that is, the crew scheduling. The duties must meet several requirements due to labor laws, union agreements and also the operational rules of a company. This way, the problem becomes NP-hard, for which there has been no polynomial algorithm to obtain its optimal solutions so far (Fischetti *et al.*, 1987).

A single crew scheduling is made up of a set of *tasks* and it is called a *duty*, or crew daily duty. A task is a set of trips, of the same vehicle, which must be performed by the same crew. Each task has deterministic starting and ending times, and also starting and ending point. The set of all tasks, provided by vehicle scheduling, must be assigned to a set of duties with minimum cost. The set of all duties composes a complete crew scheduling, or for short, a crew scheduling. Duties can be classified into two groups: single duty or split duty. In the single duty the tasks are done straight and the time intervals between the tasks are less than a given value. In this case, the idle time between tasks is counted for remuneration. If a single interval longer than the given value occurs, the duty is classified as split duty. This kind of duty is associated to busses operating only during the peaks hours existing on workdays timetable and the interval between two pieces of duty is not counted for remuneration. On the other hand, split duties must appear in reduced amount in the solution, limited to 20% of the total of duties.

In order to assign the tasks and build the duties, several operational constraints and labor rules must be taken into account. In this case, the following restrictions were considered: *i*) each task must be assigned to one duty; *ii*) each duty is a sequence of tasks that can be performed

by a single crew, without overlapping; *iii*) the normal work time of a duty is of 6 hours and 40 minutes; *iv*) the single duties must have a break of at least 20 minutes for rest and meal; *v*) the duties cannot do more than two hours overtime; *vi*) the time interval between the end of a duty and its start on the next day must have at least 11 hours; *vii*) the change of crews, during operation, can happen only at predetermined places and times; *viii*) a duty can do a maximum number of vehicle changes during the day.

The main goal of a solution to the problem is to reduce the total number of duties, and the total of overtime payment. At the same time, the number of split duties must be kept to a lower level, being limited to 20% of the number of duties. This problem can be placed as a multiobjective optimization, since it brings two conflicting objectives: to minimize the number of duties and the amount of overtime payment. In this work the cost function for the CSP was defined as a linear combination of the fixed cost, representing crew wage, and variable costs, given by overtime payment and the number of split duties. Therefore, the CSP is treated as a mono objective minimization, while satisfying all the constraints mentioned above.

2. Literature Review

The most widely used approach to deal with this problem models it as a set covering or a set partitioning problem. The strategy of column generation is largely used to solve the problem, as can be seen in the works of Smith and Wren (1988), Desrochers and Soumis (1989), Fores *et al.* (1999) and Barnhart *et al*. (1998). However, exact models are limited in practical applications, since they are unable to solve very large problems. So, it is necessary to use heuristic methods to solve problems that appear in real life, which are large.

One of the pioneer groups in this area, named Scheduling and Constraint Management Group of Leeds University, carried out a set of heuristic implementations using Genetic Algorithms (Kwan *et al.*, 1999; Li and Kwan, 2003), Tabu Search (Shen and Kwan, 2001), Ant System (Forsyth and Wren, 1997) among others. The models developed by this group are widely used in the United Kingdom to build crew scheduling as well as the scheduling of the operational fleet (Wren, 2004).

Although the Crew Scheduling Problem has been widely studied and applied in more developed countries, its solving techniques are not disseminated and rarely applied to the Brazilian reality. Partially, this happens because the companies do not have the necessary input data and organization, and also because there is a lack of models and commercial systems that represent the Brazilian operational reality.

Among the studies concerning the CSP solved in the Brazilian reality, we can highlight metaheuristic models that use Simulated Annealing, Tabu Search, GRASP and VNS (Silva *et al.*, 2002; Soares *et al.*, 2006; Souza *et al.*, 2004; Reis and Silva 2012). These models have been tested with data of companies operating in Brazilian public transport system and the results show that there are great possibilities to reduce costs in relation to the solutions adopted by the companies. On the other hand, new rules for the problem and modern search techniques have appeared in the past few years, which can be employed to solve this problem. Thus, this work explores the use of a recent local search technique based on a graph representation of the problem and the use of network flow algorithms to carry out more complex local search than those inherent to the classic local search procedures.

The local search technique named Very Large-scale Neighborhood Search (VLNS) (Ahuja, 2000) was first employed to solve CSP by Silva and Cunha (2010). In this work, VLNS was used as local search procedure of GRASP metaheuristic. It was observed by the authors that the performance of VLNS search is strongly dependent on the initial solution. Therefore, further works propose to adopt single-solution-based metaheuristics that make periodical perturbations in the current solution through different movements. In this class of metaheuristics, the Variable Neighborhood Search (VNS) (Mladenović and Hansen, 1997) and Iterated Local Search (ILS) (Lourenço *et al.*, 2010) can be pointed out. The VNS metaheuristic consists in exploring the space of solutions through systematic changes of neighborhood structures, while the main idea of ILS is iteratively to perturb the obtained local optimal and to apply a local search to this perturbed solution. The metaheuristic VNS was already implemented making use of VLNS to solve the CSP. The details and results can be found in the work from Reis and Silva (2012).

In this context, this work aims to solve the Crew Scheduling Problem using the metaheuristic ILS combined with the VLNS search technique. In order to verify the efficiency of the proposed combinations, the ILS was also implemented in their classical version, using First

Improvement Method as local search. Both ILS versions were tested with real data from a company that operates in a public transportation system, and the results were compared with the results obtained by VNS using the code developed by Reis and Silva (2012).

3. Local Search Procedures

The ILS metaheuristic was implemented in the classical version, which uses the First Improvement Method as local search. The ILS-VLNS, version uses the Very Large-scale Neighborhood Search as local search procedure. Both versions from ILS were tested by solving large scale problems of Brazilian reality.

3.1 Evaluation Function

The cost C_s associated with a solution s of the CSP is computed by means of a linear combination of the fixed cost and the variable costs. The fixed cost represents the wage of a crew, and the variable cost is the total of overtime payment. Finally, the split duty crew is weighted so that the user may have a control on the number of duties of this type into the solution. The final expression for the cost of a solution is:

$$C_s = \sum_{i=1}^{tot\_crews} (Fix\_Cost + w_1 \times overtime_i + w_2 \times split\_duty_i) \quad (1)$$

In Expression (1), *Fix_Cost* represent the fixed remuneration of a crew, w_1 is the weight per minutes of overtime and $overtime_i$ is the total of overtime into duty i, expressed in minutes. And w_2 is the weight for split duty and $split\_duty_i$ is equals 1 if crew i is a split duty and 0 otherwise. The weight w_2 is calibrated to control the number of split duties in the solution, since this kind of duty is not desirable in large amount.

3.2 Initial Solution

The initial solution was built according to the manual procedure, which can be seen as a greedy method of tasks allocation. In the procedure, the first duty is initialized with the first task of a bus. The procedure goes on allocating the next task from the same bus which does not superpose the previous one and that generates the shortest idle time. The duty is

completed as soon as it presents overtime and does not exceed maximum working time allowed. This procedure is repeated while there is a task not assigned. Algorithm 1 presents the logic employed to build an initial solution for the CSP.

Algorithm 1 - Procedure Initial Solution

Procedure Initial Solution (solution *s*, set of busses with its tasks**)**
begin
1. for each vehicle do //called current bus
2. while there are tasks from current bus not assigned do
3. initialize a new duty in *s*; //called current duty
4. while the current duty is not full and there are tasks not assigned in the current bus do
5. assign the first task not assigned to the current duty, such that do not superpose the tasks already allocated to him;
6. end while
7. end while
8. end for
9. return *s*;
end;

3.3 Classical Methods of Local Search

Methods of local search are algorithms based on the concept of *neighborhood*. So, consider ***S*** the space of all solutions of a optimization problem and ***f*(.)** the function to be minimized. Let us define a kind of move ***m*** as being a modification that changes one solution ***s*** in other solution ***s'***, called *neighbor* of ***s*** by move ***m***, and there are so many different *neighbors* as feasible moves of type ***m*** applied to solution *s*. The function ***N*** that depends on the neighborhood structure of the problem, associates to each solution ***s*** in ***S***, its neighborhood ***N*(*s*)** contained in ***S***. Each solution ***s'*** in ***N*(*s*)** is called of *neighbor* of ***s*** according to a kind of neighborhood structure that is defined by its move ***m***. For instance, let us define the move *task exchange* for the CSP. Then, the neighborhood ***N*(*s*),** from a given solution ***s***, consists of all solutions produced from ***s*** by exchanging two tasks from different duties, generating feasible new duties. Broadly, local search heuristics start from an initial feasible solution, and it walks

from *neighbor* to *neighbor* according to the adopted *neighborhood* structure, keeping the best solution visited during the procedure.

3.3.1 Relocation-Exchange neighborhood structure

Once the maximum number of tasks involved in a move is defined, it is necessary to establish different kinds of moves which characterize a neighbor of a solution. Two kinds of moves were adopted: *task relocation* and *task exchange*, both between two duties and without superposing. These moves are performed to find the best neighbor of a current solution. As an example, for a given $k \leq 3$, consider two duties i and j, randomly obtained. Then, k consecutive tasks are randomly picked out of duty i to be introduced into duty j. Thus, one of the following situations may occur: 1) the k tasks of i can be introduced in j, without the necessity of removing any task of j. In this case the movement is accepted and the new solution will be evaluated. 2) The introduction of k tasks in j demands the removal of one or more tasks from this duty. In this case, if the tasks removed from j can be inserted in duty i, without any superposing with the remaining tasks in i, the movement is accepted, otherwise it is rejected. In both cases, the changes are considered if and only if the resulting duties are feasible, i.e. if the duties do not violate any constraints of the problem.

In this paper, the classical descend method called Random Descent Method (Mladenović and Hansen, 1997) was adopted, which besides avoiding exhaustive exploitation of the neighborhood, analyzes the duties in a random order, renewed at each iteration. It prevents the first duties of the schedule from present higher quality in detriment to the quality of the last duties.

3.4 Very Large-scale Neighborhood Search to solve the CSP

A critical aspect in neighborhood search algorithms is the choice of the neighborhood structure, that is, the way it is defined. This choice largely defines whether the search strategy will lead to solutions of good quality or not. In general, the larger the neighborhood the better the quality of the local optimal solution shall be. However, large-scale neighborhoods require a long research time. For this reason, a larger neighborhood does not imply in a better heuristic, unless the neighborhood is explored efficiently.

Such algorithms are called Very Large-scale Neighborhood Search Methods, applicable to Set Partitioning Problems - SPP (Ahuja *et al.*, 2000). These algorithms enable to explore very large neighborhoods, while keeping the processing time at very low levels. One way to achieve such efficiency is using network flow algorithms to enumerate implicitly a neighborhood, in order to find better solutions.

A *cyclic exchange* may be defined as a sequence of elements t_1-t_2-t_3-…-t_r-t_1 belonging to different subsets of a partitioning from SPP. Consider a solution $S = S_1, \ldots, S_p$, i.e., a partition for the set of all tasks $t_k \in T$, where each partition S_i represents a duty. Without loss of generality, let $S[t_i]$ be the subset to which the element t_i belongs, then the cyclic exchange t_1-t_2-t_3-…-t_r-t_1 represents the exchange where the element t_1 is moved from $S[t_1]$ to $S[t_2]$, the element t_2 from $S[t_2]$ to $S[t_3]$, and so on. Finally, the element t_r is moved from $S[t_r]$ to $S[t_1]$. - Figure 1 shows a cyclic exchange t_1-t_2-t_3-t_1 before and after the task exchange. A path exchange is defined by a sequence of nodes t_1-t_2-t_3-…-t_r and differs from the cyclic exchange by the fact that the last element t_r is moved from $S[t_{r-1}]$ to $S[t_r]$ and no element is moved from $S[t_r]$ to $S[t_1]$. In Figure 2 there is an example of path exchange. Observe that with the tasks exchange subset S_1 has one task less and subset S_3 has one task more than before the exchange. In order to implement path exchange it is necessary to introduce additional nodes and arcs.

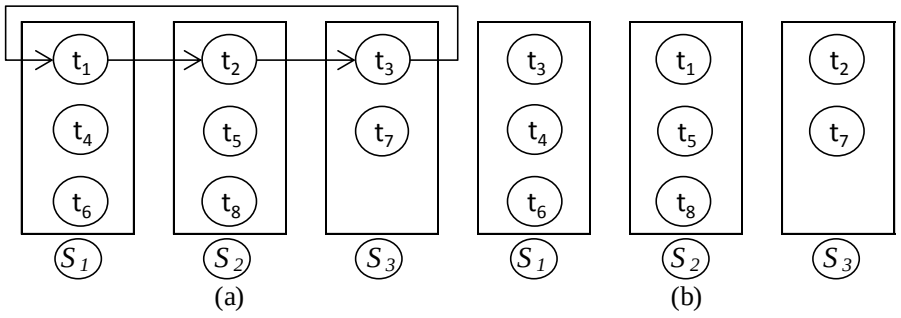

Figure 1 - Cyclic exchange (t_1-t_2-t_3-t_1) before (a) and after the exchange (b)

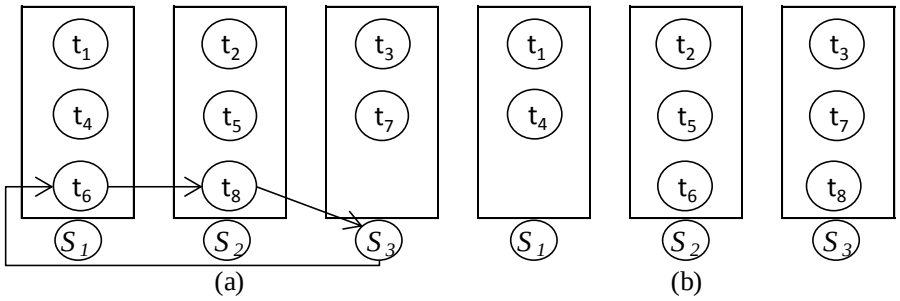

Figure 2 - Path exchange (t_6-t_8) before (a) and after the exchange (b)

The classical neighborhood search methods are based on relocation and two-exchange moves of elements between the two subsets to which they belong. Observe that the neighborhood of cyclic exchange and path exchange contemplates the two-exchange and still explore an infinity of other solutions unreachable by the classical relocation and two-exchange moves. Therefore, it is expected that the local optimal solutions obtained by multiple exchanges are, on average, superior to those obtained by two-exchange moves. However, once the size of the neighborhood in multiple exchanges increases exponentially with the size of the problem, it is necessary to have an efficient method to find a solution of lower cost in the neighborhood. This problem can be overcome using the concept of *improvement graph* and network flow algorithms to explore efficiently a given neighborhood.

An improvement graph for a neighborhood with multiple exchanges is defined for a feasible solution S for the problem, being represented by $G(S)$. Let $S[t_j]$ be the duty that contains task t_i. The graph $G(S)$ is a directed graph with n node, where node i corresponds to a task $t_i \in T$. A directed arc (i, j) in $G(S)$ means that task t_i leaves its current duty and it is moved to the duty containing task t_j, that is, duty $S[t_j]$. Simultaneously, task t_j leaves $S[t_j]$. To construct $G(S)$ all tasks pairs t_i and t_j in T are considered. The arc (i, j) is added to $G(S)$ if the tasks t_i and t_j belong to different duties, and the new duty composed by $\{t_i\} \cup S[t_j] \setminus \{t_j\}$ is feasible. The cost c_{ij} in arc (i, j) is defined as $c(\{t_i\} \cup S[t_j] \setminus \{t_j\}) - c(S[t_j])$ (Silva and Cunha, 2010).

A cycle W is called a *directed cycle* in the improvement graph $G(S)$ if the tasks in T, corresponding to the nodes from W, belong to different duties. W is defined as a valid cycle if it is a directed cycle of negative cost in $G(S)$. Thus, a valid cycle in $G(S)$ corresponds to a cyclic exchange which leads to an improvement in the value of the objective function of the

problem. This is an efficient way to search solutions that improve the objective function through multiple exchange movements. Therefore it is necessary to find valid cycles in the improvement graph ***G(S)***. A well known *modified label-correcting* algorithm that finds the minimum path from a given node source to all others nodes of the network was implemented to identify a valid cycle in this work. More details about this algorithm can be found in Ahuja *et al.* (1993).

The idea of VLNS-type algorithms consists in constructing a graph ***G(S)*** for a given solution ***S***, and finding a valid cycle ***W*** in ***G(S)*** which provides a better neighbor solution of ***S***. After making the cyclic exchange, inherent to the valid cycle, the graph is updated and a new valid cycle is sought. The search ends when the improvement graph does not present any valid cycle. The pseudo-code presented in Algorithm 2 summarizes the method.

Algorithm 2 - Pseudo-code of VLNS Procedure

Procedure VLNS (solution *s***)**
begin
1. construct the improvement graph *G*(*s*) for *s*;
2. while *G*(*s*) has valid cycles do:
3. identify a valid cycle in *G*(*s*);
4. improve the solution s due to the valid cycle changes;
5. update the improvement graph G(s);
6. end while;
7. return *s*;

end

4. Metaheuristic ILS to solve the CSP

The ILS is based on the idea that a local search procedure can be improved generating new initial solution, perturbing the current solution. In order to use an ILS algorithm, four components are needed: a) *Generate_Initial_Solution* - procedure that generates an initial solution s_0 to the problem; b) *Local_Search* - procedure that reaches a local optimal solution, called ***s''***, which is compared to the best current solution ***s***; c) *Perturbation* - procedure that modifies the current solution ***s*** reaching to an intermediary solution ***s'*** and d)

Acceptance_Criterion - procedure that decides from which solution the next perturbation will be applied. The Algorithm 3 shows the Basic ILS's pseudo-code.

Algorithm 3 - Iterated Local Search Heuristic

```
Procedure Basic_ILS();
begin
1.      s0 = Generate_Initial_Solution();
2.      s = Local_Search(s0);
3.      while (termination condition is not met) do
4.              s' = Perturbation(history, s);
5.              s'' = Local_Search(s');
6.              s = Acceptance_Criterion (s, s'', history);
7.      end
8.   return (best solution found s);
end ILS;
```

4.1 Perturbation and Acceptance Criterion

The perturbation procedure applied to the current solution in ILS metaheuristic was implemented following the *Relocation-Exchange* neighborhood previously presented, which depends on the value of ***k***. The same function, *Relocation-Exchange*, was adopted to perform both the local search and the perturbation procedure. In order to avoid the ILS entering a loop visiting the same neighbors, the perturbation is executed with the number of tasks to be relocated larger than the number of tasks considered into the local search procedure.

Into the perturbation procedure, in line 4 from Algorithm 3, the value of ***k*** is at least one unit larger than that one used into the *Local_Search*() on the next line. The parameter ***k*** is incremented as soon as the local search fails looking for a better solution. When it reaches a maximum value ***kmax***, previously defined, ***k*** receives the value 2 and the procedure continues until the termination condition is reached. The value for ***k*** used by the local search is always equal to 1.

In this implementation, a solution is accepted if and only if its cost is less than the cost of the best solution previously found. Moreover, no infeasible solution is accepted during the process.

Algorithm 4 - Iterated Local Search applied to the CSP

Procedure ILS (set of neighborhood structures N_i, i =1,…,$kmax$);
begin
1. s_0 = Generate_Initial_Solution();
2. s = Local_Search(s_0);
   k = 2;
3. **while** (termination condition is not met) **do**
4. s' = Perturbation(k , s);
5. s'' = *Local_Search*(s');
6. **if** $f(s'') < f(s)$
7. then $s = s''$ and k =2;
8. else $k = k$+1;
9. **end if**
10. **if** *(k > kmax)*
11. then *k = 2;*
12. **end if**
13. **end while**
14. **return** (best solution found s);
end ILS;

Algorithm 5 gives a high level pseudo-code for the perturbation procedure implemented into the ILS metaheuristic, where *max_changes* is manually tuned to produce a modification ranging from 20% to 40% of all duties. The *Relocate_Exchange* function was previously described in Section 3.3.1.

Algorithm 5 - Perturbation procedure

Procedure Perturbation(k, s);
begin
1. *s'* = *s*;
2. **for** (*i* = 1 to *max_changes*) *do*
3. randomly select a pair of different duties d_1 and d_2 not changed yet;
4. *s'* = Relocate_Exchange(*s', k,* d_1, d_2);
5. **end for**
6. **return**(neighbor *s'*);

end Perturbation;

5. Computational Results

The algorithms were tested by solving a set of seven problems concerning one week of work of a Brazilian company that operates in the public transportations system in the city of Belo Horizonte. The metaheuristics were implemented in C/C++ language and the tests were carried out in a PC with an Intel Core i7 processor and 8 GB RAM. The metaheuristics were performed for one hour and 10 runs were done for each problem. Despite the VNS being tested in a previous work, Reis and Silva (2012), this metaheuristic was once again executed, aiming to guarantee a best comparison with the ILS.

Table 1 contains the characteristics of the solutions adopted by the company and that are used as reference for the solutions obtained. In the following Tables, the line "OT" refers to the total of overtime (hours and minutes), "Crews" refers to the total of duties (units), "SD" refers to the total of split duties contained in the solution (units), "St. Dev." refers to the standard deviation and "EF" refers to the total cost of a solution, given by Expression (1). The value of EF can be seen as the monetary cost of a solution implemented in practice, since the weights represent the cost per unit of each component of the Expression (1).

The set of tests were performed taking into account the following weights: 10,000 for *Fix_Cost*, 4 for w_1, with the overtime expressed in minutes. The weight w_2, which refers to the split duties, received the value 600. These weights were empirically obtained aiming to

produce solutions with few drivers, a low amount of overtime hours and the number of split duties within the limit set by the company.

The weights used in the metaheuristics were applied to compute the cost from solutions adopted by the company. The main goal of the company is to reduce the total number of duties, and the total of overtime. At the same time, the number of split duties must be kept to a lower level, being limited to 20% of the number of duties and the crews could not do more than one vehicle change during the operation. Thus, it was possible to compare the solutions produced by the metaheuristics with those adopted by the company.

Table 1 - Data from solutions operated by the company

	Monday	**Tuesday**	**Wednesday**	**Thursday**	**Friday**	**Saturday**	**Sunday**
OT	86:41	85:55	106:05	120:14	108:11	54:35	26:57
Crews	134	130	149	162	155	124	68
SD	6	3	5	4	1	0	0
EF	1,364,404	1,322,420	1,518,460	1,651,256	1,576,564	1,253,100	686,468

Furthermore, the averages of all solutions obtained are presented, the AVG EF, as well as the average deviation given by (AVG EF - Best EF)/AVG EF. As much lower this percentage is, the more robust the method is (Talbi, 2009). That is, the difference among several solutions found, which count on a random factor, is not significant and the heuristic tends to produce very similar solutions. The improvements were calculated using Expression (2). Thus, negative values mean that the result obtained by the company is better than the one achieved by the metaheuristic.

$$(value_{company} - value_{metaheuristic}) \Big/ value_{metaheuristic} \quad (2)$$

5.1 Solutions obtained by VNS

Although the VNS heuristic had been previously tested in Reis and Silva (2012), in this paper its results had a little modification in relation to the first test, due to the better configuration of the PC used. Table 2 gives details of the best solutions for the Classical VNS method, and Table 3 gives the characteristics of the best solution provided by VNS-VLNS.

Table 2 - Characteristics of the Classical VNS solutions

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
OT	54:46	60:24	63:57	61:09	74:55	60:22	33:24
Crews	120	116	137	150	140	106	55
SD	23	15	22	34	30	17	10
Best EF	1,218,316	1,167,748	1,396,164	1,515,728	1,426,732	1,063,544	563,512
AVG EF	1,223,408	1,181,297	1,401,548	1,534,773	1,439,857	1,077,809	570,119
ST. DEV.	4,482.55	8,041.48	4,454.96	10,020.27	9,623.80	7,574.89	5,755.06

Table 3 - Characteristics of the VNS-VLNS solutions

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
OT	67:36	63:47	64:44	73:01	69:05	51:47	36:08
Crews	119	116	138	149	142	108	55
SD	18	13	23	25	17	15	8
Best EF	1,217,024	1,183,108	1,409,336	1,522,524	1,446,964	1,101,428	563,472
AVG EF	1,233,271	1,203,982	1,422,288	1,539,752	1,455,184	1,120,420	572,689
ST. DEV.	9,297.32	12,713.04	7,828.61	11,706.13	8,213.34	13,484.30	6,074.10

Table 4 contains a summary of the improvements reached by VNS solutions in relation to the solution adopted by the company (Table 1), calculated using Expression (2). The improvements are presented for each component from EF, and for its final value. Analyzing the final value, it is possible to conclude that the Classical VNS has more economical solutions than the VNS-VLNS. That occurs because normally the Classical VNS uses a larger number of split duties than VNS-VLNS does, then the Classical VNS gets a lower amount of overtime and crews.

Table 4 - Improvements reached by VNS in relation to the company's solution (in %)

	Heuristic	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Crews	**Classical VNS**	10.45	10.77	8.05	7.41	9.68	14.52	19.12
	VNS-VLNS	11.19	10.77	7.38	8.02	8.39	12.90	19.12
OT	**Classical VNS**	36.82	29.70	39.72	49.14	30.75	-10.60	-23.93
	VNS-VLNS	22.01	25.76	38.98	39.27	36.14	5.13	-34.08
SD	**Classical VNS**	19.17	12.93	16.06	22.67	21.43	16.04	18.18
	VNS-VLNS	15.13	11.21	16.67	16.78	11.97	13.89	14.55
EF	**Classical VNS**	10.71	**11.70**	**8.05**	**8.21**	**9.50**	**15.13**	17.91
	VNS-VLNS	**10.80**	10.53	7.19	7.80	8.22	12.10	**17.92**

5.2 Solutions obtained by ILS

The ILS metaheuristic generated results similar to those produced by VNS, as shown in Tables 5 and 6. However, comparing the Classical ILS and ILS-VLNS, the best solutions were obtained by the second method, according to the data presented in Table 7.

Table 5 - Characteristics from solutions obtained by Classical ILS

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
OT	52:33	62:35	61:58	59:20	67:58	62:35	23:57
Crews	121	116	138	151	142	106	57
SD	20	20	23	24	18	16	8
Best EF	1,234,612	1,187,020	1,408,672	1,538,640	1,447,112	1,084,620	580,548
AVG EF	1,244,630	1,204,278	1,422,502	1,554,936	1,468,522	1,102,346	589,029
ST. DEV.	8,948.39	9,031.14	10,268.90	12,587.26	10,276.23	8,226.99	6,632.47

Table 6 - Characteristics from solutions obtained by ILS-VLNS

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
OT	64:24	65:12	62:03	60:48	73:45	58:30	23:02
Crews	119	115	138	151	141	108	57
SD	18	19	20	17	16	9	7
Best EF	1,216,256	1,177,048	1,406,892	1,534,792	1,437,300	1,099,440	579,728
AVG EF	1,233,277	1,202,322	1,425,890	1,538,452	1,463,668	1,110,343	580,146
ST. DEV.	8,492.37	13,161.90	9,651.10	3,802.08	13,725.17	9,806.07	307.02

Table 7 - Improvements reached by ILS in relation to the company's solution (in %)

	Heuristic	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Crews	**Classical ILS**	9.70	10.77	7.38	6.79	8.39	14.52	16.18
	ILS-VLNS	11.19	11.54	7.38	6.79	9.03	12.90	16.18
OT	**Classical ILS**	39.38	27.16	41.59	50.65	37.17	-14.66	11.13
	ILS-VLNS	25.71	24.11	41.51	49.43	31.83	-7.18	14.53
SD	**Classical ILS**	16.53	17.24	16.67	15.89	12.68	15.09	14.04
	ILS-VLNS	15.13	16.52	14.49	11.26	11.35	8.33	12.28
EF	**Classical ILS**	9.51	10.24	7.23	6.82	8.21	**13.45**	15.43
	ILS-VLNS	**10.86**	**10.99**	**7.35**	**7.05**	**8.83**	12.26	**15.55**

In this case, ILS-VLNS produced solutions that usually have a lower number of crews and split duties, keeping the total of overtime at the same level as Classical ILS. Therefore, the value of EF was better for all solutions, losing only on Saturday.

6. Analysis of Results

The results obtained in these tests have a fewer number of split duties than those presented by Reis and Silva (2012), mainly in the case of the Classical versions. As in the previous work, the use of VLNS as local search produced solutions with fewer split duties. This behavior is observed both in VNS and in ILS metaheuristic. The algorithms produced solutions with similar characteristics and costs, despite the random features. On the order hand, metaheuristics were able to produce solutions with lower cost than the solutions adopted by the company.

Comparing the two versions from ILS, the ILS-VLNS produced better solutions, according to EF, and yet maintained the number of split duties lower than the Classical ILS. This kind of solutions is the most desirable in practice.

The comparison between the two metaheuristics, in their best version, shows that the difference in the EF value is insignificant. In this case, ILS-VLNS produces solutions with the smallest number of split duties, therefore it is the most indicated to be adopted by the company, as can be seen below.

6.1 Comparing VNS versus ILS performance

Although the Classical VNS produced better improvements than ILS-VLNS, the difference between them is very small, being insignificant in terms of percentages. The values in Table 8 were obtained by Expression (3) with data from respective rows. Negative values show the superiority of the Classical VNS over ILS-VLNS.

$$(VNS_{Classical} - ILS_{VLNS}) / VNS_{Classical} \quad (3)$$

Table 8 - Percentage difference between Classical VNS and ILS-VLNS (in %)

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
OT	-17.59	-7.95	2.97	0.57	1.56	3.09	31.04
Crews	0.83	0.86	-0.73	-0.67	-0.71	-1.89	-3.64
SD	21.74	-26.67	9.09	50.00	46.67	47.06	30.00
EF	0.17	-0.80	-0.77	-1.26	-0.74	-3.38	-2.88

By analyzing these results, it is possible to see that the difference between the Best EF for the two methods is at most 1.26% for workdays and 3.38% for the weekend. Hence, it is clear that both methods are competitive and generate close solutions for this problem. The difference between them concerns the total overtime for the workdays, which is lower for the Classical VNS. However, an important operational characteristic for the problem is the total number of split duties. Looking at this item, ILS-VLNS is superior in six out of seven days (except on Tuesday). The average deviation of split duties on workdays is 1.58 for ILS-VLNS against 7.4 for Classical VNS. An elevated number of split duties makes it hard to elaborate the monthly crew scheduling, thus this kind of duty is avoided by the company.

6.2 Statistical Inferences

Based on the results of each instance, the average, the standard deviation, the upper and lower bounds for different confidence intervals were computed. The confidence interval gives the range of values where a new result will be inside within a percentage of confidence. They were obtained using the Expression (4).

$$IC_{(1-\alpha)}(\mu) = \left[\bar{X} - t_{n-1,\frac{\alpha}{2}} \cdot \frac{S}{\sqrt{n}} \; ; \; \bar{X} + t_{n-1,\frac{\alpha}{2}} \cdot \frac{S}{\sqrt{n}}\right] \quad (4)$$

In the expression (4), we have: *(1-α) = 100(1-α)%* of confidence; $\bar{X}$ is the average of each problem (each day); $t_{(n-1,\alpha/2)}$ is the value obtained on the t-Student distribution table, according to the *n-1* degree of freedom and the value *α/2* is the desired confidence degree; *S* is the Standard Deviation of the solutions and *n* is the sample length. The t-Student distribution was used because the variance and the mean of the results are unknown.

Based on the results of each heuristic for each problem, the lower and upper bounds (LB and UB respectively), for a 90%, 95% and 99% confidence interval, were extracted and they are shown on Tables 9, 10, 11 and 12.

Table 9 - Bounds for Classical VNS

	90%		95%		99%	
	LB	**UB**	**LB**	**UB**	**LB**	**UB**
Monday	1.220.809,71	1.226.006,29	1.220.201,60	1.226.614,40	1.218.801,10	1.228.014,90
Tuesday	1.176.635,99	1.185.958,41	1.175.545,07	1.187.049,33	1.173.032,64	1.189.561,76
Wednesday	1.398.965,30	1.404.129,90	1.398.360,93	1.404.734,27	1.396.969,06	1.406.126,14
Thursday	1.528.964,99	1.540.581,41	1.527.605,62	1.541.940,78	1.524.474,96	1.545.071,44
Friday	1.434.278,80	1.445.435,60	1.432.973,22	1.446.741,18	1.429.966,43	1.449.747,97
Saturday	1.073.418,44	1.082.199,96	1.072.390,82	1.083.227,58	1.070.024,17	1.085.594,23
Sunday	566.782,91	573.454,69	566.002,16	574.235,44	564.204,09	576.033,51

Table 10 - Bounds for VNS-VLNS

	90%		95%		99%	
	LB	**UB**	**LB**	**UB**	**LB**	**UB**
Monday	1.227.881,65	1.238.659,95	1.226.620,36	1.239.921,24	1.223.715,57	1.242.826,03
Tuesday	1.196.612,54	1.211.350,66	1.194.887,87	1.213.075,33	1.190.915,90	1.217.047,30
Wednesday	1.417.750,18	1.426.825,82	1.416.688,14	1.427.887,86	1.414.242,22	1.430.333,78
Thursday	1.532.966,59	1.546.537,41	1.531.378,52	1.548.125,48	1.527.721,14	1.551.782,86
Friday	1.450.422,77	1.459.944,43	1.449.308,54	1.461.058,66	1.446.742,42	1.463.624,78
Saturday	1.112.603,89	1.128.236,11	1.110.774,58	1.130.065,42	1.106.561,64	1.134.278,36
Sunday	569.167,97	576.209,63	568.343,95	577.033,65	566.446,19	578.931,41

Table 11 - Bounds for Classical ILS

	90%		95%		99%	
	LB	**UB**	**LB**	**UB**	**LB**	**UB**
Monday	1.239.442,70	1.249.816,50	1.238.228,75	1.251.030,45	1.235.432,97	1.253.826,23
Tuesday	1.199.042,74	1.209.512,46	1.197.817,56	1.210.737,64	1.194.995,93	1.213.559,27
Wednesday	1.416.550,08	1.428.454,72	1.415.156,98	1.429.847,82	1.411.948,64	1.433.056,16
Thursday	1.547.639,45	1.562.231,75	1.545.931,84	1.563.939,36	1.541.999,17	1.567.872,03
Friday	1.462.565,83	1.474.478,97	1.461.171,74	1.475.873,06	1.457.961,10	1.479.083,70
Saturday	1.097.576,86	1.107.114,34	1.096.460,77	1.108.230,43	1.093.890,39	1.110.800,81
Sunday	585.184,72	592.873,68	584.284,94	593.773,46	582.212,74	595.845,66

Table 12 - Bounds for ILS-VLNS

	90%		95%		99%	
	LB	**UB**	**LB**	**UB**	**LB**	**UB**
Monday	1.228.354,23	1.238.199,37	1.227.202,14	1.239.351,46	1.224.548,85	1.242.004,75
Tuesday	1.194.692,76	1.209.951,24	1.192.907,20	1.211.736,80	1.188.794,98	1.215.849,02
Wednesday	1.420.296,18	1.431.484,62	1.418.986,90	1.432.793,90	1.415.971,57	1.435.809,23
Thursday	1.536.247,74	1.540.655,46	1.535.731,94	1.541.171,26	1.534.544,05	1.542.359,15
Friday	1.455.711,87	1.471.623,33	1.453.849,89	1.473.485,31	1.449.561,69	1.477.773,51
Saturday	1.104.659,15	1.116.027,25	1.103.328,84	1.117.357,56	1.100.265,10	1.120.421,30
Sunday	579.967,64	580.323,56	579.925,99	580.365,21	579.830,06	580.461,14

Based on the Lower Bounds values for the three different confidence intervals (CI), it is possible to infer that the VNS produces best lower bounds in its Classical version and the best lower bounds of ILS are associated with its VLNS version, despite the CI, and moreover, comparing Classical VNS with ILS-VLNS, the Classical VNS still has the best lower bounds to the problem. Figures 3 and 4 present a graphic comparison of lower and upper bounds, for 95% confidence interval, among the heuristics of four problems: Monday, Tuesday, Thursday and Friday. Analyzing the graphic, it is possible to reinforce the statements above.

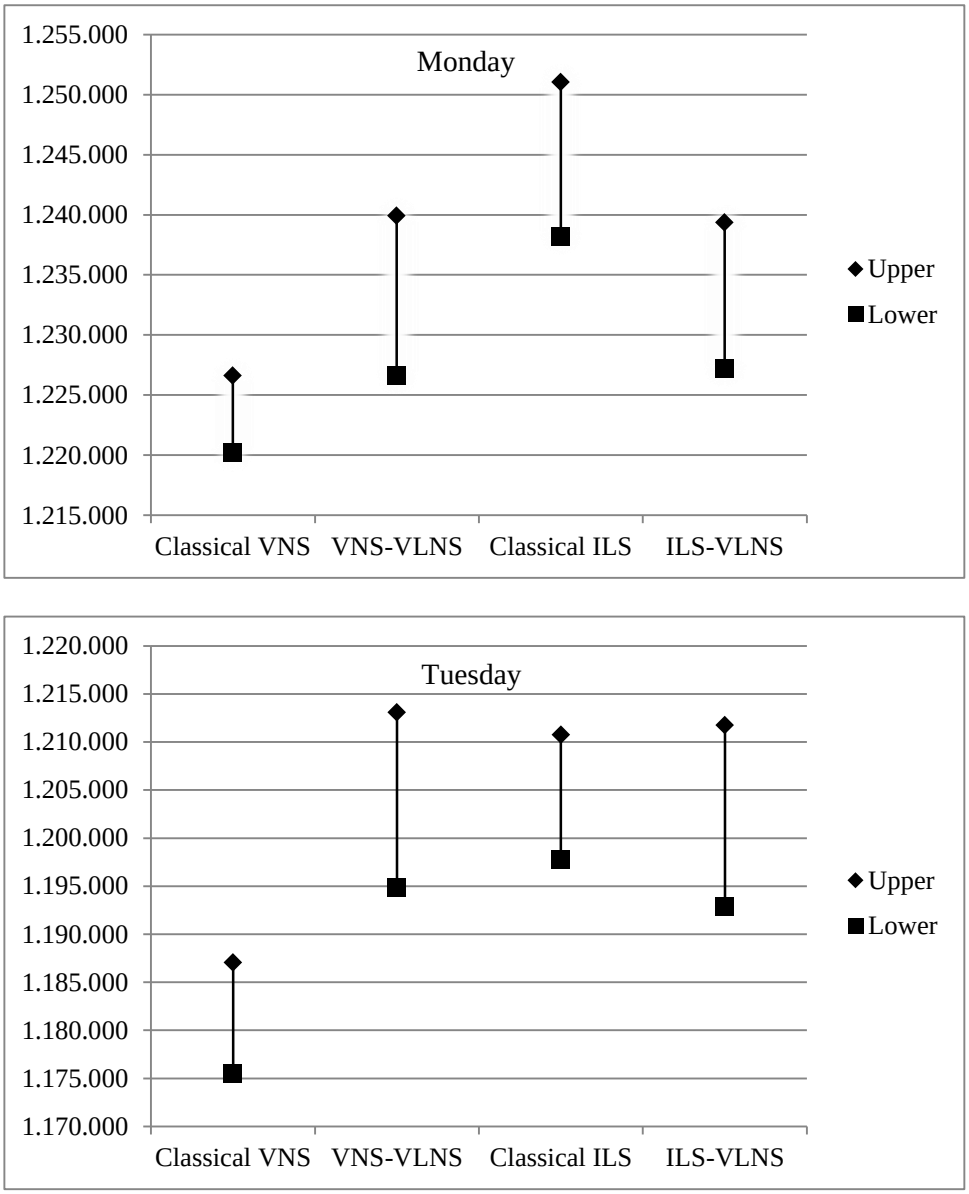

Figure 3 - Lower bounds considering 95% confidence interval of Monday and Tuesday problems

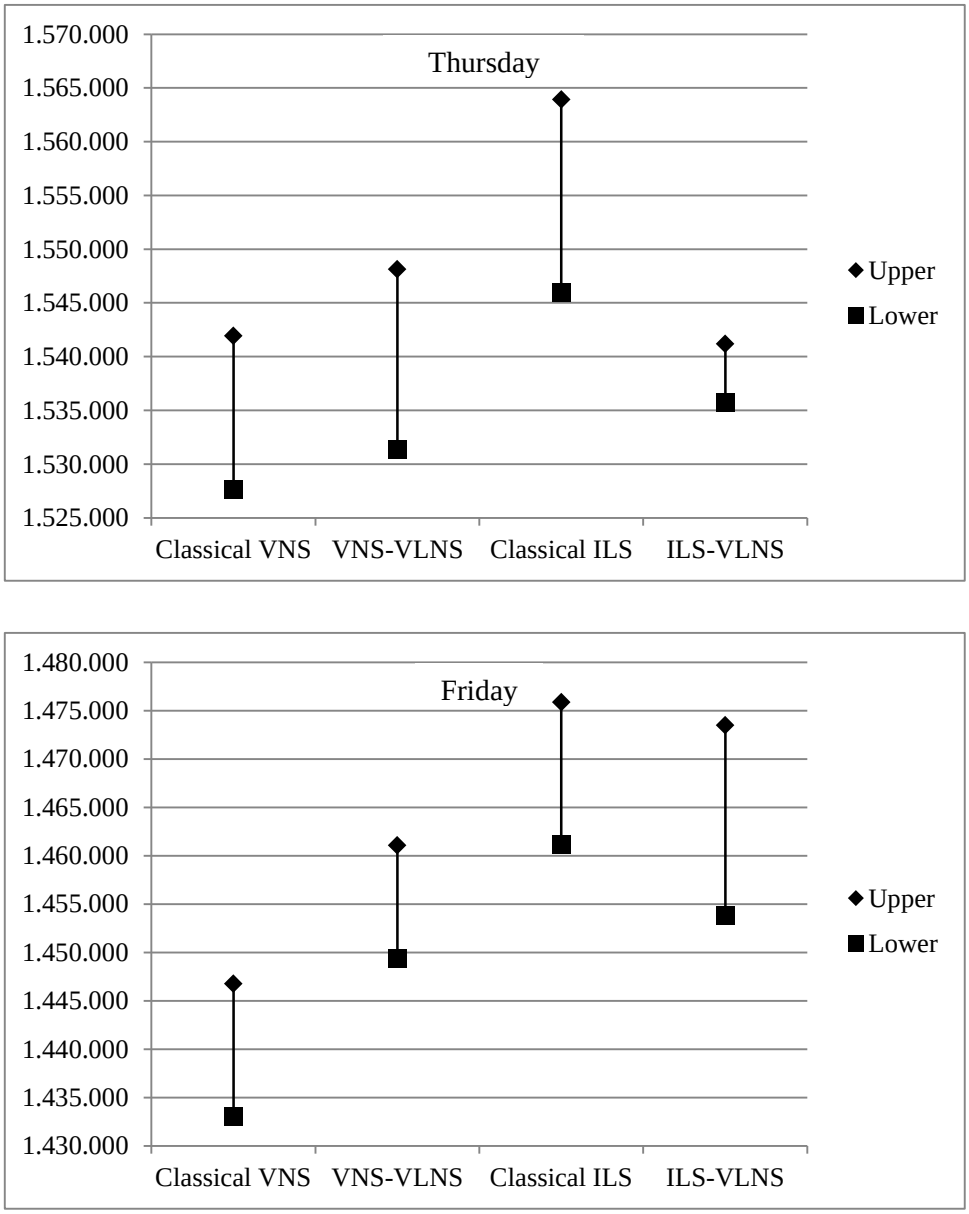

Figure 4 - Upper bounds considering 95% confidence interval of Thursday and Friday problems

Conclusion

This paper presents an implementation of ILS metaheuristic to solve the Crew Scheduling Problem of a Public Transportation System. Initially, the ILS was implemented using, as local search, the classical First Improvement method. Next, the Very Large-scale Neighborhood Search (VLNS) technique was used as local search procedure in the metaheuristics. The two versions were tested with real data from a Brazilian company that operates in the city of Belo Horizonte, Brazil, and the results were compared with those obtained by metaheuristic VNS in the classical version, and also using the VLNS local search technique.

Based on the analysis of results, it is possible to state that the use of the VLNS technique was able to outperform the classical First Improvement implementation in the ILS. It is important to point out that the VLNS version produced a smaller number of split duties than the classical local search method, showing that this search method does not cause drastic changes in the initial solution, built with the same philosophy as the company's. The combination ILS-VLNS is the one which best fits this criterion. On the other hand, the VNS with First Improvement was the best one, when comparing the value of objective function.

Further studies can be carried out including new practical restrictions in the optimization model such as limiting the number of split duties in the solution and exploring other metaheuristics with the local search technique presented in this work.

Acknowledgements

The authors thank CNPq, FAPEMIG and AUTUMN TI for their support during the development of this work.

References

Ahuja, R. K., Magnanti, T. L., Orlin, J. B. (1993) Network Flows: Theory, Algorithms, and Applications. Prentice Hall, N. J.

Ahuja, R. K., Orlin, J. B. and Sharma, D. (2000) *Very large-scale neighborhood search.* International Transactions. Operational Research, vol. 7, pp. 301-317.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. P. and Vance, P. H. (1998) *Branch-and-price: column generation for solving huge integer programs.* Operations Research, vol. 46, pp. 316-329.

Coffin, M. and Saltzman, M. J. (2000) *Statistical Analysis of Computational Tests of Algorithms and Heuristics*, INFORMS Journal on Computing, vol. 12, n. 1, pp. 24-44.

Desrochers, M. and Soumis, F. (1989) *A Column Generation approach to the Urban Transit Crew Scheduling Problem.* Transportation Science, vol. 23, pp. 1-13.

Fischetti, M, Martello, S. and Toth, P. (1987) *The Fixed Job Schedule Problem with Spread-Time Constraint*. Operations Research vol. 35, pp. 849-858.

Fores, S., Proll, L. and A. Wren. (1999) *An Improved ILP System For Driver Scheduling.* Computer-Aided Transit Scheduling, Wilson, N. H. M. (ed.), Springer, Berlin, pp. 43-61.

Forsyth, P. and Wren, A. (1997) *An Ant System for Bus Driver Scheduling.* 7th International Workshop on Computer-Aided Scheduling of Public Transport, Boston.

Kwan, A. S., Kwan, R. S. K. and Wren, A. (1999) *Driver scheduling using genetic algorithms with embedded combinatorial traits.* Computer-Aided Transit Scheduling, Wilson, N. H. M. (ed.), Springer, Berlin, pp. 81-102.

Li, J. and Kwan, R. S. (2003) *A fuzzy genetic algorithm for driver scheduling.* European Journal of Operational Research, vol. 147, pp. 334-344.

Li, J. and Kwan, R. S. (2005) *A Self-Adjusting Algorithm for Driver Scheduling.* Journal of Heuristics, vol. 11, pp. 351-367.

Lourenço, H. R., Martin, O. C. and T. Stutzle. (2010) *Iterated Local Search: Framework and Applications.* International Series in Operations Research & Management Science, vol. 146, pp. 363-397.

Mladenović, N. and Hansen, P. (1997) *Variable Neighborhood Search*, Computers & Operations Research, vol. 24, pp. 1097-1100.

Reis, A. F. S. and Silva, G. P. (2012) *Um Estudo de Diferentes Métodos de Busca e a Metaheurística VNS para Otimizar a Escala de Motoristas de Ônibus Urbano*, Transporte em Transformação XVI - Trabalhos vencedores do Prêmio CNT Produção Acadêmica 2011, pp. 45-64. CNT/ANPET.

Shen, Y. and Kwan, R. S. (2001) *Tabu search for driver scheduling.* Computer-Aided Scheduling of Public Transport, Voss, S. and J. R. Daduna (ed.), Springer, Berlin, pp. 121-135.

Silva, G. P. and Cunha, C. B. (2010) *Uso da técnica de busca em vizinhança de grande porte para a programação da escala de motoristas de ônibus urbano*. Revista Transportes, vol. 18, pp. 64-75.

Silva, G. P., Alves, J. M. C. B. and Souza, M. J. F. (2002) *Resolução do Problema de Programação Diária da Tripulação de Ônibus Urbano via Simulated Anneling.* XVI Congresso de Pesquisa e Ensino em Transportes, Panorama Nacional de Pesquisa em Transportes, vol. 2, pp. 95-104.

Smith, B. M. and Wren, A. (1988) *A Bus Crew Scheduling System Using a Set Covering Formulation*. Transportation Research, vol. 22A, pp. 97-108.

Soares, G. F., Silva, G. P. and Marinho, E. H.. (2006) *Alocação da mão de obra no Sistema de Transporte Público: Uma visão multiobjectivo.* Panorama Nacional de Pesquisa em Transportes, pp. 693-704.

Souza, M. J. F., Carvalho, L. X. T., Silva, G. P., Rodrigues, M. M. S. and Mapa, S.M.S. (2004) *Metaheurísticas aplicadas ao Problema de Programação de Tripulações no Sistema de Transporte Público*, Tendências em Matemática Aplicada e Computacional, vol. 5, n. 2, pp. 357-368.

Talbi, E. (2009) *Metaheuristics : from Design to Implementation.* John Willey and Sons.

Wren, A. (2004) *Scheduling Vehicles and Their Drivers - Forty Years' Experience.* 9th International Conference on Computer-Aided Scheduling of Public Transport, pp. 27-40.