EngOpt 2008 - International Conference on Engineering Optimization
Rio de Janeiro, Brazil, 01 - 05 June 2008.

An algorithm based on Iterated Local Search, Variable Neighborhood Descent and Tabu Search for the Integrated Vehicle and Crew Scheduling Problem

Marcone Jamilson Freitas Souza, Gustavo Peixoto Silva, Sabir Ribas, Igor Machado Coelho

Programa de Pós-Graduação em Engenharia Mineral, Departamento de Computação
Universidade Federal de Ouro Preto
Campus Universitário, 35.400-000 Ouro Preto, Minas Gerais, Brazil
{marcone, gustavo}@iceb.ufop.br, {sabirribas, igor.machado}@yahoo.com.br

Abstract

This work deals with the Integrated Vehicle and Crew Scheduling Problem (VCSP) in urban mass transit. The Vehicle Scheduling Problem (VSP) consists in creating a daily routine of operation for a fleet of vehicles of a company so that all timetabled trips are covered and the operational costs of such activity reduced, making the best use of the fleet. In the Crew Scheduling Problem (CSP) each crew corresponds to a duty, and the goal is the generation of its schedule, satisfying a set of labor agreement and operational rules of the company, as well as the workforce optimization. Traditionally, the VSP and the CSP are solved independently, e.g., by vehicles-first duties-second approach, but these methodologies do not explore the link between both problems. The difficulty of solving VCSP is greater than the isolated problems since it contains all the degrees of freedom of vehicle scheduling and all the degrees of freedom of crew scheduling. Since VCSP is NP-hard, an algorithm based on metaheuristics approaches is proposed here. This algorithm combines Iterated Local Search (ILS), Variable Neighborhood Descent (VND) and Tabu Search (TS). In order to explore the solution space, some moves based on reallocation and swap of both, trips of the vehicles schedule and tasks of the duties schedule are used. The algorithm was tested with real data provided by BHTRANS, the manager of the transit mass system of the city of Belo Horizonte, Brazil. The results show the effectiveness of the proposed approach.

Keywords: Vehicle Scheduling, Crew Scheduling, Integrated Vehicle and Crew Scheduling, Public Transport, Metaheuristics

1. Introduction

The public transportation system planning is a very complex activity. Thus it is generally divided into five main stages, which are: route planning, timetables defining, vehicle scheduling, crew scheduling and crew rostering. In general, the output of one stage is the input for the subsequent one. This work deals with the integration of two stages, the vehicle and crew scheduling problems.

According to Freling *et al.* (2003), there are three different approaches to solve the vehicle and crew scheduling: Sequential, Independent and Integrated.

In the sequential approach, the vehicle and crew scheduling problems are solved separately and the solution of one is used as input for the other. The sequential approach, in turn, can be classified as Traditional or Reverse.

The traditional sequential approach was the first one to be studied and it is widely adopted by the companies. It consists in solving the *vehicle scheduling problem* (VSP) and then, considering this solution as input, the *crew scheduling problem* (CSP) is solved. This approach has the inconvenience that the vehicle scheduling solution guides the crew scheduling solution (FREELING *et al.*, 2003), which has the most relevant cost among the ones involved in the system (BOUZADA, 2003). An alternative to solve it is to apply the traditional sequential approach, but considering the crew features during the vehicle scheduling solution. Even though the vehicle scheduling costs increase, it becomes easier to solve the CSP and much probably the total cost will be lower than the traditional approach. Scott (1985), Darby-Dowman (1988) and Reis (2006, 2007) use this strategy for solving the problem involving vehicle and crew scheduling.

In the reverse sequential approach, the CSP is solved first and then, considering this solution as input, the VSP is solved. This strategy is justified for two main motives: 1) the crew costs are higher than vehicle costs and 2) the set of vehicle restrictions is much smaller, which suggests solving CSP first, because it has fewer alternatives for solution. Bassi *et al*. (2007) propose a GRASP heuristic to solve the independent crew scheduling problem (ICSP), which builds the crew duties directly from the timetable, independently of the vehicle scheduling. In a second stage, the vehicle schedule problem is solved having as input the driver shifts already defined. A path-relinking mechanism was also implemented in order to improve the ICSP solution. The heuristic was implemented for several transport companies and the results were compared with the traditional approach. According to the authors, it can be concluded that this strategy is able to produce solutions with lower costs than the traditional process.

In the independent approach the vehicle schedule and the crew schedule are built directly from trip timetables. Therefore, in this approach the vehicle schedule is built without considering any crew characteristics and the crew schedule is obtained without considering vehicle constraints. Generally, the results obtained by this approach are not feasible in practice, because some vehicle trips will not have a driver, and some drivers will operate virtual vehicles that are not in the vehicle schedule. This approach is used to determine lower bounds for VSP and CSP.

In the integrated approach both problems are solved together in the same model. As both problems are NP-hard (GAREY and JOHNSON, 1979), then Vehicle and Crew Scheduling Problem (VCSP) is NP-hard. It is obvious than the difficult of solving VCSP is bigger because the solution space is huge, since VCSP contains all the degrees of freedom of vehicle scheduling and all the degrees of freedom of crew scheduling. However, with the improvement in the CPU speed of the computers, combined with algorithm progress, the research has taken up this topic recently. Freling *et al*. (1995), Haase and Friberg (1999) and Freling *et al*. (1999) use this type of approach. See Borndröfer *et al*. (2006) and Freeling *et al*. (2003) for a survey and experience about this topic.

This paper proposes an algorithm which combines Iterated Local Search, Variable Neighborhood Descent and Tabu Search heuristics for solving VCSP. This heuristic strategy is justified because VCSP is NP-hard and according to the literature these approaches are able to produce good solutions for each problem, considered individually (See Souza *et al*, 2007 and Marinho *et al*, 2004).

The rest of the paper is organized as follows. Section 2 describes the problem. Section 3 presents the proposed methodology, including the representation of the problem, the procedure employed for generating an initial solution, the neighborhood structure, the evaluation function and finally the adaptation of the Iterated Local Search, VND and Tabu Search heuristics to the problem. Section 4 shows the results achieved in this study and discusses the improvement obtained. Section 5 brings the concluding remarks of this work.

2. Problem definition

According to Freeling *et al*. (2003), the Vehicle and Crew Scheduling Problem (VCSP) is defined as follows. Given a set of trips within a fixed planning horizon, the objective is to find a minimum cost schedule for the vehicles and the crews, such that both the vehicle and the crew schedules are feasible and mutually compatible. Each trip has fixed starting and ending times, and the traveling times between all pairs of location are known. A vehicle schedule is feasible if (1) each trip is assigned to a vehicle, and (2) each vehicle performs a feasible sequence of trips, where a sequence of trips is feasible if a vehicle can execute each pair of consecutive trips in the sequence. Some trips do not belong to any route and these are called the ***dead trips***. A dead trip time is the time a vehicle takes to travel between two locations (terminals or depots) that do not belong to any route. This is necessary in order to move the vehicle from an ending point to the next starting point, so that the vehicle can start the next trip. A vehicle schedule defines which trips have to be performed by the same vehicle and this defines the so-called blocks. If there is enough time between one trip and the next, a change of driver may occur, and these are called ***relief points***, defined by location and time. The blocks are divided by relief points, and each part is the so-called ***task***, that is defined by two consecutive relief points and represents the minimum piece of work that can be assigned to a crew. These tasks have to be assigned to crew members, and the tasks of a crew member define a crew duty. All the duties constitute a crew schedule. Such schedule is feasible if (1) each task is assigned to one duty, and (2) each duty is a sequence of tasks that can be performed by a single crew, both from a physical and a legal point of view. In particular, each duty must satisfy several complicated constraints corresponding to work load regulations for the crews. Typical examples of such constraints are the maximum working time without a break, minimum break duration and maximum total working time. The cost of a duty is usually a combination of fixed costs such as wages, and variable costs such as overtime payment. The subsection 3.4 details the considered requirements in this work.

It is assumed that there is only one depot and all vehicles are always available.

3. Proposed methodology

In this section is presented a hybrid heuristic, which combines Iterated Local Search, Variable Neighborhood Descent and Tabu Search for reaching suboptimal solutions to the VCSP.

3.1. A solution representation to the problem

A solution s to the VCSP consists in a (s_V, s_C) pair, where s_V represents a solution to the VSP and s_C represents a solution to the CSP. The s_V solution consists in a set of vehicles, where each one has an associated list of trips to be realized during a day. The s_C solution consists in a set of crew duties, where each one is associated to a list of tasks to be carried out daily.

3.2. Construction of an initial solution

An initial solution to the VCSP is made by a sequential mechanism. To begin with, an initial solution for the vehicle scheduling is built. Later, from this vehicle solution, a crew scheduling solution is generated. Both solutions are built by a greedy heuristic.

An initial solution to the VSP is obtained by applying a constructive heuristic where at each iteration, a new trip not yet assigned is added to the current schedule of the best vehicle, in accordance to the value of the evaluation function relative to each vehicle scheduling (see eq. (2) in subsection 3.4). The choice of the trip to be added to the vehicle schedule is totally greedy, which means that the trip with lower cost will be chosen. The approach ends when all the trips are allocated. The trips belonging to each vehicle are known as the blocks of the vehicle.

From the blocks of each vehicle, a solution for the CSP is built. Initially, for each block, the trips are grouped in a task until a relief point is found. A relief point is defined by location and time, where and when a change of driver may occur.

Next, the tasks are ordered by their starting time. At each step, there is a set of non-empty duties and a duty without tasks. The constructive procedure works as follows. The first task is assigned to the first duty, the empty one. Next, for each task of the sequence, we evaluate its insertion for all the duties and also for a new duty without tasks. The evaluation of each crew duty is carried out according to the eq. (8), relative to the crew scheduling (see subsection 3.4). The task is assigned to the duty with the

lowest cost. The crew schedule stops when all the tasks are allocated.

3.3. Neighborhood Structure

With a view to defining the neighborhoods of a given solution *s* six moves were applied. The N_V^R, N_V^S, N_C^R and N_C^S neighborhoods use moves based on reallocation and swap of both, trips of the vehicles schedule and tasks of the crew duties schedule, respectively. The N_V^{RP} and N_V^{SP} neighborhoods consist in reassigning and swapping the trips of vehicles followed by the reconstruction of the duties affected by the moves according to the previous subsection, respectively.

3.4. Evaluation function

The cost function of VCSP is computed by the eq. (1):

$$f(s) = \alpha \times f_V(s) + \beta \times f_C(s) \tag{1}$$

where $f_V(s)$ represents the component of *f* that evaluates *s* with respect to the vehicle scheduling (see eq. (2)), $f_C(s)$ evaluates *s* with respect to the duty scheduling (see eq. (8)) and α and β are weights.

$$f_V(s) = \sum_{k \in F} f_v^k(s) + \omega_{CR} \times \mathbf{C}hangeRoutes + \omega_{ESB} \times ExcSplitBlocks \tag{2}$$

In the eq. (2), $f_V^k(s)$ is the cost of the vehicle *k* of the fleet, obtained through the eq. (3), *ChangeRoutes* is the total number of changes of routes made by all the vehicles, ω_{CR} is the penalty associated to it, *ExcSplitBlocks* represents the excess of times that each vehicle returns to the depot and stays there more than 120 minutes and, finally, ω_{ESB} is the respective penalty. By operational restrictions, we observe that the maximum number of times that it can occur is 60% of the fleet size.

$$f_V^k(s) = \sum_{(i,j) \in T} C_{ij} + CD_{d1} + CD_{nd} \tag{3}$$

In the eq. (3), *T* is the set of *n* trips of the vehicle *k*, C_{ij} is obtained by the eq. (4) and represents the cost to perform the consecutive trips *i* and *j*; CD_{d1} is the transportation cost of the vehicle *k* from the depot to the its first trip and CD_{nd} is the transportation cost of the vehicle *k* between its last trip and the depot.

$$C_{ij} = \begin{cases} CD_{ij} + CC_{ij} & if\ \ t_{ij} < 0 \\ CT_{ij} + CD_{ij} & if\ \ 0 \le t_{ij} \le MaxTimeTerm \\ CD_{id} + CD_{dj} & if\ \ t_{ij} > MaxTimeTerm \end{cases} \tag{4}$$

In the eq. (4), CT_{ij}, CD_{ij} and CC_{ij} represent, respectively, the time that the vehicle stays at the terminal between trips *i* and *j*, the time of a dead trip and the time in minutes that a trip *i* coincides with a trip *j*. In the eq. (4), *MaxTimeTerm* is the maximum time that a vehicle can stay at the terminal, established according to the operational politics of the Public Transport System and t_{ij} is the time, in minutes, that the vehicle is idle (i.e., without productive activity) between the trips *i* and *j*. Therefore t_{ij} is the time between the beginning of the trip *j* and the ending of the trip *i* minus the time of the dead trip if this one is necessary to reposition the vehicle.

The CD_{ij} and CT_{ij} costs are obtained as follows:

$$CD_{ij} = \begin{cases} \left(\alpha_1 \times dt_{ij}\right)^{\beta_1} & if\ \ dt_{ij} \le TimeLimitDeadTrip \\ dt_{ij} \times \omega_{TD} & if\ \ dt_{ij} > TimeLimitDeadTrip \end{cases} \tag{5}$$

$$CT_{ij} = \begin{cases} \left(\alpha_2 \times t_{ij}\right)^{\beta_2} & if\ \ t_{ij} \le TimeLimitTerm \\ t_{ij} \times \omega_{TT} & if\ \ t_{ij} > TimeLimitTerm \end{cases} \tag{6}$$

where: dt_{ij} is the time of dead trip between the trips *i* and *j*; *TimeLimitDeadTrip* means the maximum time of the company schedule in which a dead trip can be done, w_{TD} is the weight admitted if the dead trip time is greater than *TimeLimitDeadTrip*, *TimeLimitTerm* is the maximum time of the company schedule that a vehicle stays at the terminal; w_{TT} is the weight considered if the vehicle stays at the terminal more than *TimeLimitTerm* minutes, $\alpha_1 \in [0, 1]$ and $\beta_1 \in [0, 1]$ are weights utilized to adjust the costs in eqs. (5) and (6), respectively.

The conflict CC_{ij} between trips *i* and *j* is evaluated by the eq. (7), as follows:

$$CC_{ij} = -(t_{ij}) \times \omega_{TC} \tag{7}$$

where w_{TC} is the weight associated to this infeasibility.

Let $R = \{r_1, r_2, \ldots, r_{|R|}\}$ be the set of routes of the company and $S_1, S_2, \ldots, S_{|S|}$ subsets of R, called groups of routes, with $R = S_1 \cup S_2 \cup \ldots \cup S_{|S|}$ and $S_1 \cap S_2 \cap \ldots \cap S_{|S|} = \varnothing$.

The evaluation of the duty scheduling is given by the eq. (8), which must be minimized:

$$\begin{aligned} f_C(s) = \sum_{i \in Duties} \big(& PenaltyIdleTime_i + PenaltyOvertime_i + \\ & \mu_3 \times ChangeVehicles_i + \mu_4 \times ChangeRoutes_i + \mu_5 \times TerminalChange_i + \\ & \lambda_1 \times ConflictTime_i + \lambda_2 \times ExcTime_i + \lambda_3 \times DutyTimeInsuf_i + \lambda_4 \times ChangeVehiclesFor_i + \\ & \lambda_5 \times ChangeRoutesFor_i + \lambda_6 \times TerminalChangeFor_i \big) + \lambda_7 \times ExcessSplitDuty + \delta \times nDuties \end{aligned} \quad (8)$$

where:

(i) *Duties* represents the set of duties;

(ii) μ_i and λ_i are the penalties applied to the constraints and objectives, respectively;

(iii) $PenaltyIdleTime_i$ assumes the value μ_1 if the idleness of the crew on the duty i is greater than two hours and $0.347 \times IdleTime_i^{0.8}$, otherwise, where $IdleTime_i$ corresponds to the time that each crew is idle on the duty i;

(iv) $PenaltyOverTime_i$ assumes μ_2 when the overtime is greater than 120 minutes and $1.357 \times Overtime_i^{0.6}$, otherwise, where $Overtime_i$ indicates the time, in minutes, that the crew i is in overtime payment;

(v) $ChangeVehicles_i$ indicates the number of times that the crew is reassigned to a different vehicle during the duty i and there is enough time to do the changes;

(vi) $ChangeRoutes_i$ indicates the number of times that the crew assigned to a duty i in the route r_j is reassigned to a another route r_k , and both routes must belong to the same group, i.e., $r_j \in S_m \subset R$ and $r_k \in S_m$;

(vii) $TerminalChange_i$ indicates the number of times that the crew is reassigned to a different terminal during the duty i. This crew reassignment is allowed in the following situations: (1) if the terminal involved belongs to the same group of terminals; (2) if the involved tasks belong to the same vehicle and (3) if there is at least 120 minutes of idleness between two consecutive tasks of the duty i, that is, in a split duty, a crew reassignment to a different terminals is always allowed. On the other hand, a crew reassignment is forbidden if the reallocation does not satisfy none of the three items previously described;

(viii) $ConflictTime_i$ represents the total time, in minutes, with respect to the duty i, that there are tasks in conflict, for example, two consecutive tasks of the same crew, when the second starts before the first ends;

(ix) $ExcTime_i$ represents the time, in minutes, with respect to the duty i, that surpasses nine hours of daily work. The worked time in each duty i is computed as follows: $WorkedTime_i$ = (final time of the last task of the duty i) – (the initial time of the first task) – (the duration of the split, if the duty is classified as split duty);

(x) $DutyTimeInsuf_i$ = max {0, 11×60 – (1440 – time in which duty i ends + time in which duty i begins)}. It occurs because between two consecutive labor days must be 11 hours of resting. Therefore, this factor represents the amount of time, in minutes, in respect to the duty i, in which this labor law is not verified;

(xi) $ChangeVehiclesFor_i$ indicates the number of times that the crew is reassigned to a different vehicle during the duty i and there are not enough time to do the changes;

(xii) $ChangeRoutesFor_i$ indicates the number of times that a crew allocated to a route r_j in a duty i is also allocated to another route r_k not belonging to the same group, i.e., $r_j \in S_m \subset R$ and $r_k \notin S_m$;

(xiii) $TerminalChangeFor_i$ represents the number of times during the duty i that there are forbidden reassignments in relation with terminals (See item (vii));

(xiv) *ExcSplitDuty* represents the number of split duties in the solution that is greater than 1.25 × (fleet size). It represents an estimative of the company that assumes that at most 50% of the crews are necessary to realize split duties.

(xv) *nDuties* represents the number of duties and δ is the penalty applied to minimize the number of duties in the schedule;

3.5. The ILS-VND-TSAR algorithm

With an aim to introduce the Variable Neighborhood Descent – VND (Hansen and Mladenovic, 2003), let N^k (k=1,…,k_{max}) be a finite set of pre-selected neighborhood structures, and $N^k(s)$ the set of solutions in the kth neighborhood of s. Neighborhoods N^k may be induced from one or more metric (or quasi-metric) functions introduced into a solution space S. An *optimal solution* is a feasible solution where a minimum of the evaluated function f is reached. There is a *local minimum s'* of f with respect to $N^k(s)$, if there is no solution $s \in N^k(s')$ such that $f(s) < f(s')$. Metaheuristics based on local search procedures try to continue the search by other means after finding the first local minimum. VND is based on three simple facts: (1) A local minimum with respect to one neighborhood structure is not necessary so with another; (2) A global minimum is a local minimum with respect to all possible neighborhood structures; and (3) For many problems local minimum with respect to one or several are relatively close to each other.

According to these authors, this last observation, which is empirical, implies that a local optimum often provides some information about the global one. This may for instance be several variables with the same value in both. However, it is usually not known which ones are such. An organized study of the neighborhood of this local optimum is therefore in order, until a better one is found.

The Variable Neighborhood Descent (VND) approach is obtained if a change of neighborhoods is performed in a deterministic way

and its main steps are presented in Figure 1.

Procedure VND	
1:	Input: s, $f(.)$,the set of neighborhood structures N^k, for $k = 1, \ldots, k_{max}$, that will be used in the search
2:	$k \leftarrow 1$
3:	**while** $k \leq k_{max}$ **do**
4:	Find the best neighbor $s' \in N^k(s)$
5:	**if** $f(s') < f(s)$ **then**
6:	$s \leftarrow s'$
7:	$k \leftarrow 1$
8:	**else**
9:	$k \leftarrow k + 1$
10:	**end if**
11:	**end while**
12:	**return** s

Figure 1 – Basic VND procedure

For large instances, finding the best neighbor at each iteration (line 4 of Figure 1) is quite costly. In these cases, a random approach is normally used and this is what is done here. A classical Random Descent (RD) procedure works as follows. A point *s'* is generated at random from the *k*th neighborhood of *s* (line 4 of Figure 1). If *s'* is better than the incumbent, then *s* is updated to *s'* ($s \leftarrow s'$) and the search continues from *s*. If there are not any improvements in *DescentMax* iterations, then the search stops. Therefore, the procedure can return not necessarily a local optimum. In case of improvement, the search starts from the first neighborhood ($k \leftarrow 1$). This mechanism is justified because the neighborhoods are nested such that N^{k+1} is more computationally complex than N^k.

The proposed procedure to do a local search in the solution space of the VCSP, called VND_2_LEVELS, is described in the Figure 2. This procedure is based on the ideas of the VND heuristic in the sense that not only the neighborhoods are changed, but also the local search procedures. In this procedure, s_0 is a resulting solution of the perturbation in a local optimum. The RDVVP (s_0, N_V^{RP}) is the Random Descent Procedure that makes use of reallocation movements belonging to the N_V^{RP} neighborhood structure, and the tasks of the crews are allocated in a greedy way, ignoring the weights of the crew scheduling, that is, $\beta = 0$ in equation (1). The RDVVP (s_1, N_V^{SP}) is the Random Descent Algorithm that makes swapping movements of the N_V^{SP} neighborhood, ignoring the weights of the crew scheduling function. The RDVVI (s_4, N_V^{RP}) and RDVVI (s_5, N_V^{SP}) are similar to the RDVVP (s_0, N_V^{RP}) and RDVVP (s_1, N_V^{SP}), respectively, but in these cases the weights of the crew scheduling function are computed. These random approaches are interrupted if no improvements are found in *DescentMax* iterations. The TSAR_R and TSAR_S are algorithms based on Tabu Search metaheuristic with Adaptive Relaxation (Glover, 1996) that makes use of the neighborhoods N_C^R and N_C^S, respectively. These algorithms consist in changing the weights of the infeasibilities periodically during the search, either promoting the generation of infeasible solutions or feasible solutions in order to explore other regions not yet visited in the solution space. These algorithms were used by Marinho *et al*. (2004) to solve the Crew Scheduling Problem (CSP) and a detailed description can be found in their work.

Procedure	**VND_2_LEVELS**
Inicialization	Let s_0 be a initial solution
Step 1	$s_1 \leftarrow$ RDVVP (s_0, N_V^{RP}) $s_2 \leftarrow$ RDVVP (s_1, N_V^{SP}) If the value of s_2 is lower than s_0, do $s_0 \leftarrow s_2$ and return to the beginning of the Step 1
Step 2	$s_3 \leftarrow$ TSAR_R (s_2) $s_4 \leftarrow$ TSAR_S (s_3) If s_4 is better than s_2, do $s_2 \leftarrow s_4$ and run the Step 2 again
Step 3	$s_5 \leftarrow$ RDVVI (s_4, N_V^{RP}) $s_6 \leftarrow$ RDVVI (s_5, N_V^{SP}) If s_6 is better than s_4, do $s_4 \leftarrow s_6$ and executes again the Step 3 If there was some improvement in this step, return to the Step 2
Step 4	Return s_6

Figure 2 – VND_2_LEVELS procedure applied to the VCSP

The Iterated Local Search – ILS procedure (Lourenço, Martin and Stützle, 2003) is a metaheuristic with four basic components: *GenerateInitialSolution*, *LocalSearch*, *Perturbation* and *AcceptanceCriterion*. The *GenerateInitialSolution* is a module that consists in building a good solution to the problem. *LocalSearch* is the module that starts the search from a solution and returns a local optimum. *Perturbation* consists in realizing modifications on the local optimum in order to escape from it. Finally, the procedure *AcceptanceCriterion* consists in determining whether the new solution is accepted or not as the new current solution. According to the authors, a reasonable first guess for the acceptance criterion is to force the cost to decrease. *AcceptanceCriterion* has a strong influence on the nature and effectiveness of the walk in the space of solutions. Roughly, it can be used to control the balance between intensification and diversification of that search.

The Figure 3 presents the pseudo-code of ILS-VND-TSAR, an ILS procedure combined with VND and Tabu Search, applied to the VCSP.

Algorithm ILS-VND-TSAR

```
1:  Input: s, f(.) , LevelMax , ILSmax , TimeMax
2:  s* ← s
3:  level ← 0
4:  while level < LevelMax do
5:     iter ← 0
6:     while iter < ILSmax and time < TimeMax do
7:        s ← Perturbation(s, level)
8:        s ← VND_2_LEVELS(s)
9:        if f(s) < f(s*) then
10:          s* ← s
11:          level ← 0
12:          iter ← 0
13:       else
14:          s ← s*
15:          iter ← iter + 1
16:       end if
17:    end while
18:    level ← level +1
19: end while
20: return s*
```

Figure 3 – ILS-VND-TSAR algorithm applied to the VCSP

Six levels of perturbations have been considered. A perturbation of level *i* consists in realizing *i* + 2 random movements in the N_V^{RP} neighborhood.

4. Computational results

The proposed algorithm, so-called ILS-VND-TSAR, was implemented in C++ language using the Borland C++ Builder environment, version 5.0 and tested in a PC Intel Pentium IV, with 3.0 GHz and 1 GB of RAM running Windows XP Professional Edition.

Preliminary tests were performed to calibrate the parameters of the ILS-VND-TSAR algorithm. The number of iterations without improvement of the ILS (*ILSmax*) was fixed at 50. The number of iterations without improvement of the TSAR (*BTmax*) was fixed at 500. The *LevelMax* value was fixed at 6 and the number of iterations without improvement of the Random Descent approaches (*DescentMax*) was fixed as number of trips × number of vehicles. The total processing time was limited to half an hour. The first phase of the traditional sequential approach uses ILS algorithm described in Souza *et al.* (2007) and second one uses the TSAR algorithm described in Marinho *et al*. (2004). Both algorithms stop after 15 minutes of execution.

In order to test the algorithms, we examined scheduling problems for one public transportation company that operates in the city of Belo Horizonte, Brazil.

Tables (1) and (2) show the value of the parameters used in the evaluation function, described in section 3.4. In the Table (1), the values of the parameters *TimeLimDeadTrip* and *TimeLimTerm* are linked to each instance and represent the maximum time of a dead trip and the maximum waiting time of the vehicles at the terminal, respectively, in minutes, found in the solution utilized by the company.

Table 1 – Instances and their respective time limits with respect to the VSP

Instances	***TimeLimDeadTrip***	***TimeLimTerm***
G02_SUN_90V	18	83
GO2_MON_260V	18	82
G02_FRI_260V	18	77
G02_SAT_172V	18	75

Table 2 – Parameters of the VCSP evaluation function

Parameter	Value	Parameter	Value
MaxTimeTerm	119 minutes	μ_1	18
α_1	2	μ_2	40
β_1	ln (16) / ln (α_1 x *TimeLimDeadTrip*)	μ_3	19
α_2	1	μ_4	5
β_2	ln (16) / ln (α_2 x *TimeLimTerm*)	μ_5	5
ω_{TD}	50	λ_1	40
ω_{TT}	40	λ_2	40
ω_{TC}	80	λ_3	40
ω_{ESB}	80	λ_4	45
ω_{CR}	5	λ_5	35
δ	100	λ_6	35
α	1	λ_7	45
β	2.5		

Table 3 contains the evaluation function values of the traditional sequential and integrated approaches. It can be observed that both best and average values were reduced in all cases analyzed and the gap between these values in each instance is also small, ratifying that the proposed algorithm is robust. In addition, there was an improvement of up to 4.19% if compared to the best solution found in the sequential approach.

Table 3: Evaluation function values from the traditional sequential and integrated approaches

	Sequential Approach			Integrated Approach			
Instance	**Best Value**	**Average Value**	**Gap**	**Best Value**	**Average Value**	**Gap**	**Improvement**
G02_SUN_90V	7036	7160,35	6,47%	6725,5	6841,5	1,72%	4,41%
G02_MON_260V	22507,5	22828,55	8,26%	21087,5	21297,7	1,00%	6,31%
G02_FRI_260V	23045,5	23354,1	7,50%	21724,5	21910,1	0,85%	5,73%
G02_SAT_172V	12938,5	13260,5	8,51%	12220	12468,05	2,03%	5,55%

Table 4 presents the details from the best solutions found on each instance, in respect of the sequential and integrated approaches.

Table 4: Schedules from the sequential and integrated approaches

G02 Company	**Sequential Approach**				**Integrated Approach**			
	SUN	**MON**	**FRI**	**SAT**	**SUN**	**MON**	**FRI**	**SAT**
Number of trips	90	260	260	172	90	260	260	172
Number of vehicles	11	40	40	23	11	40	39	23
Vehicle waiting time (hh:mm)	16:13	22:26	5:56	21:02	13:44	8:12	22:14	21:02
Dead trip time (hh:mm)	8:24	12:50	12:14	16:48	9:50	15:06	14:58	18:36
Number of split blocks	3	21	20	5	5	24	24	8
Number of split duties	12	50	50	16	7	49	43	20
Number of crews	21	62	63	37	19	62	63	34
Crew idleness (hh:mm)	15:13	15:04	9:23	15:44	11:07	7:52	6:28	15:12
Overtime payment (hh:mm)	5:04	6:44	7:34	20:08	14:58	8:28	12:31	19:44
Change of routes	1	5	7	1	4	4	6	2
Change of terminals	2	18	19	9	3	11	8	10
Change of vehicles	6	62	66	11	2	34	34	15
f_V	841	2510	2553	1491	893	2740	2887	1585
f_C	2478	7999	8197	4579	2333	7339	7535	4254
$f = f_V + 2.5 \times f_C$	7036	22507.5	23045.5	12938.5	6725.5	21087.5	21724.5	12220

From Table 4 it can be verified that the proposed algorithm was able to reduce the number of crews (From 21 to 19 on Sunday, from 37 to 35 on Saturday), the waiting time of the vehicles at the terminals (From 16:13 to 15:31 on Sunday, from 22:26 to 21:12 on Monday, from 5:56 to 2:24 on Friday and from 21:02 to 20:23 on Saturday), the crew idleness on Sunday (From 15:13 to 10:47) and the number of changes of terminals on Monday and Friday (From 18 to 13 on Monday and from 19 to 14 on Friday). It can be

observed that the number of crews grew up on Friday, but on the other hand there was a reduction on the overtime payment and on the number of changes of routes and terminals. Soares *et al.* (2006) has shown that there is a trade-off between, on the one hand, the number of trips plus crew idleness and, on the other hand, the overtime payment. This trade-off can be handled by manipulating the weights of the evaluation function. Also, it is important to observe that the integrated approach produces worst vehicle schedule, but its crew schedule are better than the ones produced by the sequential approach.

7. Conclusions

This paper dealt with the Integrated Vehicle and Crew Scheduling Problem. In order to solve it, an algorithm based on Iterated Local Search, Variable Neighborhood Descent and Tabu Search with Adaptive Relaxation, so-called ILS-VND-TSAR, is proposed. It is an adaptation of the ILS algorithm described in Souza *et al.* (2007) for solving the vehicle scheduling problem and the TSAR algorithm developed by Marinho *et al*. (2004) for the crew scheduling problem. Two new neighborhoods are introduced, which consist in reassigning and swapping the trips of vehicles followed by the reconstruction of the duties affected by the moves.

A comparison between the integrated approach and the traditional sequential one is performed. The first considers both problems simultaneously, while the later first solves the vehicle scheduling and then solves the crew scheduling. The results showed that the integrated approach was able to produce better results, with global improvement of up to 4.19%. In addition, it can be verified that the proposed algorithm is robust since the maximum variation of the final solutions is quite small, namely, 1.96%.

It is important to point out that the integrated approach produced a worst vehicle schedule, but its crew schedule was better than the traditional sequential one. This fact clearly illustrates that producing a good quality solution to the vehicle scheduling may not necessarily lead to a good quality solution to the crew scheduling.

8. Acknowledgements

The authors thank FAPEMIG (grant TEC 679/06), CNPq (grant 474831/2007-8) and the Federal University of Ouro Preto for the support given to the development of this study, as well as to BHTRANS for providing the data.

9. References

1. Bassi, H. V., Silva, G. P., Marinho, E. H., Souza, M. J. F. (2007) GRASP and Path-relinking Applied to the Independent Crew Scheduling Problem, In: International Conference on Computers in Urban Planning and Urban Management, 2007, Foz do Iguaçú, Brazil, *Reviewed papers in the Conference Proceedings of CUPUM 2007*, v. 1, p.1 – 11.
2. Borndröfer, R.; Grötschel, M. and Pfetsch, M. (2006) Public transport to the fORe!, *ORMS Today*, v. 33(2), p. 30-40.
3. Bouzada, C.F. (2003) *Costs of the Public Transport System by Bus* (in portuguese), Editora C/Arte, Belo Horizonte, Brazil.
4. Darby-Dowman, K., Jachnik, J. K.; Lewis, R. L. and Mitra, G. (1988) Integrated decision support systems for urban transport scheduling: Discussion of implementation and experience, in J. R. Daduna and A. Wren (eds.), *Computer-Aided Transit Scheduling: Proceedings of the Fourth International Workshop*, Springer Verlag, Berlin, p. 226-239.
5. Feo, T. A., Resende, M. G. C. (1995) Greedy randomized adaptive search procedures, *Journal of Global Optimization*, v. 6, p. 109-133.
6. Freeling, R. C.; Boender, G. E. and Paixão, J. M. P. (1995) An integrated approach to vehicle and crew scheduling, Technical Report 9503/A, Econometric Institute, Erasmus University Rotterdam, Rotterdam.
7. Freling, R., Huisman, D., Wagelmans, A. (2003) Models and Algorithms for Integration of Vehicle and Crew Scheduling, *Journal of Scheduling*, v. 6(1), p. 63–85.
8. Freeling, R.; Wagelmans, A. P. M. and Paixão, J. M. P. (1999) An overview of models and techniques for integrating vehicle and crew scheduling, in N. H. M. Wilson (ed.), Computer-Aided Transit Scheduling, Springer Verlag, Berlin, p. 441-460.
9. Garey, M. R., Johnson, D. S. (1979) *Computers and Intractability*, In W. H. Freeman (ed.), A Guide to the Theory of NP-Completeness, New York.
10. Glover, F. (1986) Future paths for integer programming and artificial intelligence, *Computers and Operations Research*, v. 13(5), p. 533-549.
11. Glover, F. (1996) Tabu search and adaptive memory programming – advances, applications and challenges, In R. Barr, R. Helgason, J. Kennington (Eds.), *Interfaces in Computer Science and Operations Research*, p. 1-75.
12. Glover, F., Kockenberger, G. (2003) *Handbook of Metaheuristics*, Kluwer Academic Publishers.
13. Glover, F., Laguna, M. (1997) *Tabu Search*, Kluwer Academic Publishers.
14. Haase, K. and Friberg, C. (1999) An exact branch and cut algorithm for the vehicle and crew scheduling problem, in N. H. M. Wilson (ed.), *Computer-Aided Transit Scheduling*, Springer Verlag, Berlin, p. 63-80.
15. Hansen, P.; Mladenovic, N. (2003) Variable neighborhood search: Principles and applications, *European Journal of Operations Research*, v. 130, p. 449-467.
16. Lourenço, H. R.; Martin, O.; Stützle, T. (2003) Iterated Local Search, In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, Kluwer Academic Publishers.
17. Marinho, E. H., Ochi, L. S., Drummond, L. M. A., Souza, M. J. F. and Silva, G. P. (2004) Tabu Search Applied to the Bus Driver Scheduling Problem (in portuguese), In Simpósio Brasileiro de Pesquisa Operacional, 2004, São João Del Rey, Brazil, *Proceedings of the XXXVI SBPO*, Rio de Janeiro: SOBRAPO, 2004, v. 1. p. 1471 – 1482.
18. Reis, J. v. A. (2007) *Integrated Vehicle and Bus Driver Scheduling* (in portuguese) Exame de qualificação de mestrado, Programa de pós-graduação em Engenharia de Transportes, São Paulo: Escola Politécnica da Universidade de São Paulo, Brazil.
19. Reis, J. v. A. (2006) *Optimization techniques applied to the daily vehicle and crew scheduling* (in portuguese) Monografia de graduação em Engenharia de Produção, Departamento de Computação, Ouro Preto, Brazil: Universidade Federal de Ouro Preto, 70 p.

20. Soares, G. F., Silva, G. P. Marinho, E. H., Simões, E. L., Souza, M. J. F. (2006) Optimization in the Public Transport System (in portuguese). *Proceedings of the 16th Simpósio de Pesquisa Operacional e Logística da Marinha – SPOLM*, Escola de Guerra e Mar, Brazil.
21. Scott, D. (1985) A large linear programming approach to the public transport scheduling and cost model, in J. M. Rousseau (ed.), *Computer Scheduling of Public Transport 2*, North Holland, Amsterdan, p. 473-491.
22. Souza, M. J. F., Silva, G. P., Simões, E. M. L. (2007) Vehicle Scheduling: an heuristic approach (in portuguese) In: *Transporte em transformação XI: trabalhos vencedores do prêmio CNT de Produção Acadêmica 2006*. Brasília: Positiva, 2007, v. 1, p. 39-57.