A hybrid metaheuristic algorithm for the Integrated Vehicle and Crew Scheduling Problem

M.J.F.Souza*,[1], S.Ribas**,[2], I.M.Coelho**,[3]

Federal University of Ouro Preto, Department of Computer Science, Ouro Preto, Minas Gerais, Brazil 35400-000

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 sequentially by the 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 individual 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 with Adaptive Relaxation (TSAR). In order to explore the solution space, we defined six movements based on relocation and swap of both, trips of the vehicles schedule and tasks of the duties schedule. The algorithm was tested with real data of a Brazilian city. The results show the effectiveness of the proposed approach.

Key words: 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.

Figure 1 shows the traditional stages of public transportation system planning, where the output of one stage is the input for the subsequent one.

*Principal corresponding author
**Corresponding author
[1]marcone@iceb.ufop.br
[2]imcoelho@iceb.ufop.br
[3]sabir@iceb.ufop.br

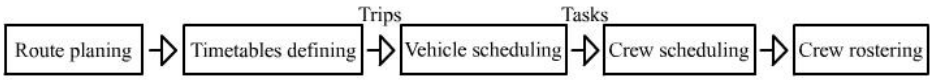

Figure 1: Stages of public transportation system planning

This work deals with the third and fourth 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 (Freling 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 et al. (1988), Reis (2008) 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.

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 Freling et al. (2003) for a survey and experience about this topic.

This paper proposes an algorithm, named ILS-VND-TSAR, 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 structures, the evaluation function and finally the adaptation of the Iterated Local Search, Variable Neighborhood Descent 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 Freling 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. We assumed too that from 05:00 to 08:00 and from 16:00 to 19:00 two minutes are necessary for a change of driver, while in the other periods, only one minute is necessary.

3. Proposed methodology

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 done 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. Figure 2 shows an example.

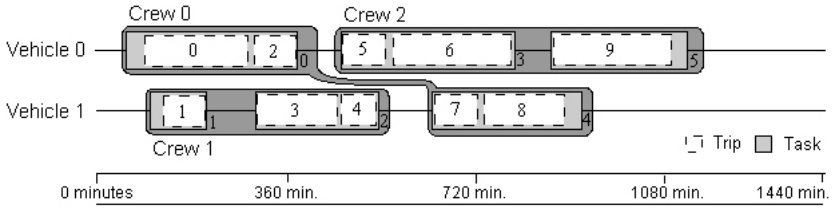

Figure 2: Solution representation

In the Figure 2 there are two vehicles and three crew duties. The Vehicle 1 performs the trips 1, 3, 4, 7 and 8. The crew 0 performs the tasks 0, makes a vehicle change and performs the task 4. Just remembering, some trips are grouped in the same task because there is not enough time to make a vehicle change.

We observe that the solution representation adopted allows a vehicle to make more than one trip at the same moment, and a crew to make more than one task at the same time. To eliminate these practical inconsistencies, our evaluation function penalize these occurrences.

3.1. Evaluation function

The cost of VCSP is calculated by equation (1), that we try to minimize. In this equation, $f_v(s)$ represents the component of f that evaluates the solution s with respect to the vehicle scheduling and $f_c(s)$ evaluates the solution s with respect to the duty scheduling.

$$f(s) = f_v(s) + f_c(s) \tag{1}$$

3.1.1. Vehicle schedule evaluation

In this work we try to minimize the follow items about the vehicle schedule:

- Number of vehicles
- Down time, i.e., the time that a vehicle is idle
- Dead trip time, i.e., the time that a vehicle performs trips without passengers

According to the companies that we contacted, a solution can not be used if more than 60% of the vehicles comes back to the depot during its duties. Thus, a vehicle scheduling is feasible if:

- There are no conflict time in the vehicle schedule

- The number of split vehicle duties is less than 60% of the number of vehicle duties

The vehicle schedule cost is given by equation (2):

$$f_v(s) = f(s_v) = \sum_{i \in Q_v} \alpha_i q_i^v(s_v) + \sum_{i \in O_v} \gamma_i o_i^v(s_v) \tag{2}$$

where Q_v and O_v are the quality and the operational requirements of the vehicle schedule, respectively; α_i and γ_i are the penalties applied to the quality and the operational requirements of a vehicle solution s_v, respectively; and q_i^v and o_i^v are the values of the ith quality and operational vehicle requirement, respectively.

3.1.2. Crew evaluation

In relation to the crew schedule, we try to minimize the follow items:

- Number of crew duties
- Overtime payment
- Crew idleness time

According to the Brazilian labor laws, the crews can not work more than nine hours in one day. Moreover, there can be no less than eleven hours between the end of a workday and the beginning of the next day journey. We also consider the following operational restriction: a solution can not be used if more than 25% of the crew comes back to the depot during its duties. Thus, a crew scheduling is feasible if:

- There is no conflict time in the crew schedule
- There is no crew working more than 9 hours per day
- There is no duty having less than 11 hours between its ending time and its next starting time
- The number of split crew duties is less than 25% of the number of crew duties

The crew schedule cost is given by equation (3):

$$f_c(s) = f(s_c) = \sum_{i \in Q_c} \beta_i q_i^c(s_c) + \sum_{i \in O_c} \theta_i o_i^c(s_c) \tag{3}$$

where Q_c and O_c are the quality and the operational requirements of the crew schedule, respectively; β_i and θ_i are the penalties applied to the quality and the operational requirements of a crew solution s_c, respectively; and q_i^c and o_i^c are the values of the ith quality and operational crew requirement, respectively.

Table 1: Used data on the weights calculation

Description	Value
Driver payment (month)	R$ 1000,99
Ticket collector payment	R$ 600,59
Crew payment	R$ 1601,58
Crew payment + taxes (38%)	R$ 2210,18
Crew workdays in a month	24
Work time during a day (minutes)	430
Payment about one work minute	R$ 0,21
Payment of one overtime minute	R$ 0,32
Payment of one workday	R$ 92,09
Fixed cost of one vehicle	R$ 13415,58
Vehicle workdays in a month	30
Vehicle cost daily	R$ 447,19
Average velocity (km/h)	20
Cost for kilometer	R$ 2,64
Cost of one minute of dead trip	R$ 0,88
Variable cost of one kilometer	R$ 0,78
Cost of one minute of downtime	R$ 0,26

3.1.3. Defining weights to the evaluation function

The objective of the evaluation function is to evaluate the quality of a solution. As we saw in previous sections, this function considers requirements that must to be minimized and some of these requirements are relatively more important than others. To contemplate the relative importance of the requirements, our the weight used are based on real costs of a Brazilian city company, as used in Reis (2008). The Table 1 presents the data used to calculate the evaluation function weights, where R$ means Brazilian currency.

The Table 2 presents the weights of the VCSP evaluation function. The quality weights are based on the real costs presented in the Table 1, while the operational weights are empirical and its values are greater than the quality ones. The last column of this table give us the characteristics of an example solution.

The equations (4), (5), (6), (7), (8) and (9) show how is made the evaluation the solution of the Table 2.

$$f(s) = f_v(s) + f_c(s) = \sum_{i \in Q_v} \alpha_i q_i^v(s_v) + \sum_{i \in O_v} \gamma_i o_i^v(s_v) + \sum_{i \in Q_c} \beta_i q_i^c(s_c) + \sum_{i \in O_c} \theta_i o_i^c(s_c) \tag{4}$$

$$\sum_{i \in Q_v} \alpha_i q_i^v(s_v) = 447.19 \times 5 + 0.26 \times 1700 + 0.88 \times 161 = 2819.63 \tag{5}$$

$$\sum_{i \in O_v} \gamma_i o_i^v(s_v) = 4000 \times 0 + 1000 \times 0 = 0 \tag{6}$$

$$\sum_{i \in Q_c} \beta_i q_i^c(s_c) = 92.09 \times 11 + 0.21 \times 1702 + 0.32 \times 815 = 1631.21 \tag{7}$$

Table 2: Weights of the VCSP evaluation function

Type	Refers to	Requirement	Weight	Example of s
Quality	Vehicles	Number of vehicles	447.19	5
Quality	Vehicles	Down time	0.26	1700
Quality	Vehicles	Dead trip time	0.88	161
Quality	Crews	Number of crews duties	92.09	11
Quality	Crews	Crew idleness time	0.21	1702
Quality	Crews	Overtime payment	0.32	815
Operational	Vehicles	Conflict time	4000	0
Operational	Vehicles	Excess of split duties	1000	0
Operational	Crews	Conflict time	4000	0
Operational	Crews	Excess of split duties	1000	0
Operational	Crews	Insufficient time between duties	4000	0
Operational	Crews	Excess time worked	4000	0

$$\sum_{i \in O_c} \theta_i o_i^c(s_c) = 4000 \times 0 + 1000 \times 0 + 4000 \times 0 + 4000 \times 0 = 0 \tag{8}$$

$$f(s) = 2819.63 + 1631.21 = 4450.84 \tag{9}$$

3.2. Constructing 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 3.1.1). 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. (3), relative to the crew scheduling (see subsection 3.1.2). The task is assigned to the duty with the lowest cost. The crew schedule is interrupted when all the tasks are allocated.

3.3. Neighborhood Structure

The heuristic methods used in this work are based on making modifications in a given solution to improve the quality. These modifications are so-called movements and we can apply different types of movements in a solution. A neighborhood structure is defined by a type of movement.

A neighbor of a given solution s is defined by applying a movement in the solution s. Applying different movements the heuristic methods are able to explore different regions of the solution space.

With a view to explore the solutions space we define six neighborhood structures, which are based on relocation (R) and swap (S) of both, trips of the vehicles (v) schedule and tasks of the duties (c) schedule are used.

Two of them (N_v^R and N_v^S) do modifications only on the vehicles, ignoring the repercussions of these movements in the crew scheduling, which creates, possibly infeasible solutions.

Two other structures (N_c^R and N_c^S) only modify the crew scheduling, preserving the compatibility between vehicles and crews.

The third pair of neighborhood structures (N_v^{RP} and N_v^{SP}) is integrated and modifies both items, vehicles and crews. These last two structures are to relocate or to change the vehicles trips followed by a reconstruction of the duties to make the schedules of vehicles and crews compatible. For each vehicle involved in the modification, all tasks that belong to these vehicles are removed from the crew scheduling. Then, from the new configuration of the blocks of the involved vehicles, the new tasks are generated. These tasks, in turn, are added to any crew member, in accordance with the greedy procedure described in the previous section. For these two structures, N_v^{RP} and N_v^{SP}, the change made is "spread" for the crews.

3.4. The Proposed Algorithm

To solve the VCSP we propose the ILS-VND-TSAR algorithm, which is described in section 3.4.5. The following sections show the components of this algorithm.

3.4.1. The Vehicle Random Descent Procedure

For large instances, finding the best neighbor at each iteration is a costly operation. In these cases, a random approach is normally used. A classical Random Descent (RD) procedure works as follows. A point s' is generated at random from the kth neighborhood of s. 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 $RDmax$ iterations, then the search stops. Therefore, the procedure can return not necessarily a local optimum.

When it comes to integrated problems, finding the best neighbor at each iteration is a much more costly operation. This is due to the fact that in integrated neighborhoods considered (N_v^{RP} and N_v^{SP}), each modification in the vehicle schedule is reflected in the crew scheduling. To reduce the number of spreadings of these modifications in the crew scheduling, the Vehicle Random Descent (VRD) procedure is proposed as described in Algorithm 1.

The working principle of this algorithm is quite simple and it is similar to the classic random descent. The main difference is that in this new algorithm, only

the changes made in vehicles (relocation or swap) that improve the evaluation function f_v are propagated in the crew scheduling.

In the Algorithm 1, s is a solution formed by the pair of solutions s_v and s_c, $VRDmax$ is the maximum number of iterations without improvement, N_v is an specific neighborhood structure (relocation or swap) for the vehicles, m is the movement made in the considered structure, N_v^P is the neighborhood structure (relocation or swap) with the spreading in accordance to the movement m and f_v is the function that evaluates the part s_v of the solution s.

It is observed that the final solution of this algorithm can, not necessarily be better than the initial solution in relation to the integrated function f since the vehicles evaluation function f_v is the only one used.

Algorithm 1: RDP

Input: Solution s, Integer $VRDmax$, Neighborhood $N_v(.)$, Neighborhood $N_v^P(.)$, Function $f_v(.)$
Output: Solution s^* better than or equal to s according to the function f_v

```
iter ← 1;
while (iter ≤ VRDmax) do
    m ← randomMovement(s, N_v);
    s' ← neighbor(s, m);
    if s' better than s according to the function f_v then
        s ← neighbor(s, N_v^P, m);
        iter ← 0;
    end
    iter ← iter + 1;
end
s* ← s;
return s*;
```

3.4.2. The Integrated Vehicle and Crew Random Descent Procedure

This strategy is similar to the one presented in the previous section. There are two basic differences. The first one is that only the movements that produce better solutions with respect to vehicles are candidates (better than f_v) or worse than f_v in at most $(\lambda - 1) \times 100\%$.

The second difference is that it considers both the costs of s_v and the s_c, which is not the case of the Vehicle Random Descent that only considers the cost of s_v when applied to solving the VCSP. The objective of this algorithm is refining a solution s using integrated movements without unduly burdening the processing, as may happen if the refinement was done by classical Random Descent algorithm using only integrated movements.

The Algorithm 2 presents the Integrated Vehicle and Crew Random Descent Procedure (IRD). In this algorithm, s is a solution formed by the pair of the mutually compatible solutions s_v and s_c, $IRDmax$ is the number of iterations without improvement in f, m is a movement made in the vehicles in accordance with the neighborhood N_v (relocation or swap), N_v^P is the neighborhood structure (relocation or swap), with spreading in accordance with the modification m, f_v is the function that evaluates s_v and λ, with $\lambda \geq 1$, is a value used to

determine if it worth processing the most expensive part of the refinement, the construction and the spreading of a integrated movement, in the absence of improvement. For example, $\lambda = 1.01$ implies that if a movement generates a solution s'_v which costs more than 1% of the cost of s_v, the method does not test the movement of spreading. Otherwise, the test is performed and, if the resulting solution s' of the movement of spreading is better than s, then the solution s is updated, that is, $s \leftarrow s'$. Tests have shown that for the instances examined, 1.01 is a good value for λ and this is the value used in this work.

Algorithm 2: IRD

Input: Solution s, Integer $IRDmax$, Neighborhood $N_v(.)$, Neighborhood $N_v^P(.)$, Function $f(.)$, Real λ
Output: Solution s^* better than or equal to s according to the function f

```
iter ← 1;
while (iter ≤ IRDmax) do
    m ← randomMovement(s, N_v);
    s' ← neighbor(s, m);
    if f_v(s') ≤ λ × f_v(s) then
        s'' ← neighbor(s, N_v^P, m);
        if s'' better than s according to the function f then
            s ← s'';
            iter ← 0;
        end
    end
    iter ← iter + 1;
end
s* ← s;
return s*;
```

3.4.3. The Tabu Search with Adaptative Relaxation Procedure

The Tabu Search methods (Glover and Kockenberger, 2003) is an iterative procedure for solving combinatorial optimization problems, which accepts non-improvement movements to avoid be trapped in optima local.

Starting from an initial solution s_0, the method explores, at each iteration, a subset V of the neighborhood $N(s)$, of current solution s. The member $s' \in V$ with the best evaluation according to the function $f(s')$ became the current solution, even if s' is worst than s. This strategy, however, can led the search process to visit solutions already reached before. In order to prevent cycles, short term memory is employed, usually in the form of a Tabu List, which forbids undoing the last movements. Movements remain in the tabu list for a given number of iterations (tabu tenure). Since this strategy can be too restrictive, in order to not disregard high quality solutions, movements with tabu status can be accepted if the produced solution satisfies an aspiration criterion, which usually considers the cost of the best solution found so far, that is, if the produced solution is better than the best solution found so far, the movement can be accepted. With the aim of reducing the risk of revisiting the same solutions, we use a dynamic tabu list, in our case, the tabu tenure value is selected from an interval, which

defines a set of possible values $\{minTabuTenure, \cdots, maxTabuTenure\}$.

The procedure TSAR (Tabu Search with Adaptive Relaxation), proposed in Marinho et al. (2004), is applied for solving the Crew Scheduling Problem. Its pseudo-code is presented at Algorithm 3.

In the TSAR procedure we seek for the First Improvement in a subset of the neighborhoods N_c^R or N_c^S, or if there is not a improvement, it returns the best neighbor in this subset. This procedure also incorporates an Adaptive Relaxation (AR) mechanism. This mechanism consists 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.

Our AR implementation is based on the proposal of Schaerf (1996), where weights of each source of infeasibility are dynamically updated, as originally proposed by Gendreau et al. (1994). For each source of infeasibility i, the corresponding weight θ_i in the objective function f_c is multiplied by a factor δ_i, which is updated as follows:

1. In the beginning of the search set $\delta_i \leftarrow 1$.
2. At every k movements:
   - if all k visited solutions are feasible considering constraint i then $\delta_i \leftarrow \delta_i/\gamma$;
   - if all k are infeasible considering constraint i then $\delta_i \leftarrow \delta_i \times \gamma$;
   - if part of k solutions are feasible, and another part is infeasible considering constraint i, δ_i keeps unchanged.

The parameter μ is randomly selected in the interval $[1.8, 2.2]$. As observed in Schaerf (1996), this strategy tends to avoid deterministic ratios which could bias the search.

Each value of δ_i is bounded by two constants $\delta_{i,min}$ and $\delta_{i,max}$, which prevents the adaptive relaxation from indefinitely increasing/decreasing weights for constraints which are always unsatisfied/satisfied.

In the Algorithm 3, *UpdateIteration*() indicates when the dynamic weights must be updated (In our case, every $k = 10$ iterations). In our implementation, the subset V consists in 50% of the number of crews, selected at random.

3.4.4. The Two Levels Variable Neighborhood Descent

With an aim to introduce the Variable Neighborhood Descent - VND (Hansen and Mladenovic, 2001; Hansen et al., 2008), 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.

Algorithm 3: TSAR

Input: Solution s, Integer *TStime*, Neighborhood N_c (N_c^R or N_c^S), Function $f_c(.)$

Output: Solution s^* better than or equal to s according to the function f_c

```
s* ← s;
TabuList ← ∅;
repeat
    Define subset V ⊆ N_c(s)
    bestMovement ← RandomMovement(V)
    bestCost ← ∞
    for all Movement m ∈ V do
        if f_c(s ⊕ m) < f_c(s*) then
            bestMovement ← m
            break
        end
        else
            if m ∉ TabuList then
                if f_c(s ⊕ m) < f_c(s) then
                    bestMovement ← m
                    break
                end
                else
                    if DynamicWeightsCost(s ⊕ m) < bestCost then
                        bestMovement ← m
                        bestCost ← DynamicWeightsCost(s ⊕ m)
                    end
                end
            end
        end
    end
    s ← s ⊕ bestMovement
    if f_c(s) < f_c(s*) then
        s* ← s
    end
    UpdateTabuList()
    if UpdateIteration() then
        UpdateDynamicWeights()
    end
until TStime reached ;
return s*
```

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.

The proposed procedure to do a local search in the solution space of the VCSP, called Two Levels Variable Neighborhood Descent (VND2L), is described in the Algorithm 4.

Algorithm 4: VND2L

Input: Solution s, Function $f(.)$, Function $f_v(.)$, Function $f_c(.)$
Output: Solution s^* better than or equal to s according to the function f

$s_0 \leftarrow s$;
$s_1 \leftarrow \text{VRD}(s_0, VRDmax, N_v^R, N_v^{RP}, f_v)$;
[$Step1$]
$s_2 \leftarrow \text{VRD}(s_1, VRSmax, N_v^S, N_v^{SP}, f_v)$;
if *s_2 better than s_0 according to the f_v* **then**
  $s_0 \leftarrow s_2$ and go back to $Step1$;
end
$s_3 \leftarrow \text{TSAR}(s_2, TStime, N_c^R, f_c)$;
[$Step2$]
$s_4 \leftarrow \text{TSAR}(s_3, TStime, N_c^S, f_c)$;
if *s_4 better than s_2 according to the function f_c* **then**
  $s_2 \leftarrow s_4$ and go back to $Step2$;
end
$s_5 \leftarrow \text{IRD}(s_4, IRDmax, N_v^R, N_v^{RP}, f, \lambda)$;
[$Step3$]
$s_6 \leftarrow \text{IRD}(s_5, IRDmax, N_v^S, N_v^{SP}, f, \lambda)$;
if *s_6 better than s_4 according to the f* **then**
  $s_4 \leftarrow s_6$ and go back to $Step3$;
end
if *there is an improvement in the Step3* **then**
  Go back to $Step2$;
end
$s^* \leftarrow s_6$;
return s^*;

The algorithm 4 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. There are two levels because this procedure has intern levels and an extern one.

The intern levels consist in three VND applications. The first one has the objective of improve the vehicles; the second one, to improve the crews and, the last one, to improve both, vehicles and crews.

The extern level is an application of VND considering that each intern level is an iteration of the external part of the VND2L.

In this algorithm, s, as we will see in the section 3.4.5, is a resulting solution

of the perturbation in a local optimum. The VRD is the Vehicle Random Descent procedure described in the section 3.4.1. The IRD is the Integrated Vehicle and Crew Random Descent Procedure presented in the section 3.4.2. These random approaches, VRD and IRD, are interrupted if no improvements are found in $VRDmax$ and $IRDmax$ iterations, respectively. The TSAR is the Tabu Search with Adaptive Relaxation procedure, described in the section 3.4.3.

3.4.5. The proposed algorithm for solving VCSP

The proposed algorithm for solving VCSP is based on Iterated Local Search (ILS) metaheuristic. ILS (Lourenço et al., 2003; Stützle, 2006) 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 these 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. In our case, a perturbation of level i consists in making $i + 2$ random movements in a given neighborhood structure.

The Algorithm 5 presents the ILS-VND-TSAR algorithm applied to solve the VCSP. In this algorithm, s is the initial solution; *ILStime* defines the time limit; $ILSmax$ is the maximum number of iterations without improvement in the same level and f is the function to be minimized. We observe that whenever there are some improvement in the current solution, the parameter *level* returns to its first value.

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 proposed algorithm. The number of iterations without improvement of the ILS (ILSmax) was fixed at 50, as well as the number of iterations without improvement of IRD (IRDmax). The number of iterations without improvement of VRD (VRDmax) was fixed at 100. The time limit to apply TSAR was fixed at 20 seconds. The total processing time was limited to 20 minutes.

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 10 minutes of execution. In this approach, *ILSmax* was fixed at 50, and one perturbation of level i é made applying $i + 2$ moves at the neighborhood N_v^R, the local search of ILS uses two nestled random descents, the first using the neighborhood N_v^R and the second, N_v^S.

Algorithm 5: ILS-VND-TSAR

```
Input: Solution s, Integer ILStime, Integer ILSmax, Function f(.)
Output: Solution s* better than or equal to s according to the function
        f
s* ← VND2L(s, f, f_v, f_c);
level ← 0;
while ILStime not satisfied do
    iter ← 0;
    while iter < ILSmax and ILStime not satisfied do
        s' ← perturbation(s*, level);
        s'' ← VND2L(s', f, f_v, f_c);
        if s'' better than s* according to the function f then
            s* ← s'';
            level ← 0;
            iter ← 0;
        end
        else
            iter ← iter + 1;
        end
    end
    level ← level + 1;
end
return s*;
```

In order to test the algorithms, we examined twelve scheduling problems for nine public transportation company that operates in a Brazilian city. The data are available at http://www.decom.ufop.br/prof/marcone/pt/instances/VC.zip.

Tables 1 and 2 show the value of the parameters used in the evaluation function, described in section 3.1.3.

Table 3 contains the evaluation function values of the traditional sequential and integrated approaches. The two first columns "Best" show the best value for f found in 20 runs of the algorithms, while the two first columns "Avg." indicate the average value in these runs. Column "'Gap'' means the gap between the average value and the best known value to the respective instance. The last columns "Best" and "Avg." indicate the improvement of the integrated approach over the sequential one with respect to the best and average solution found for each algorithm, respectively. It can be observed that the proposed algorithm was able to reduce both best and average values, as well the gap between the average value and the best known value to each instance. In addition, the gap is also small, ratifying that the proposed algorithm is robust, and there was an improvement of up to 1.64% if compared to the best solution found in the sequential approach and up to 2.39% if compared to the average one.

Tables 4 to 7 present some characteristics of the best solutions found on each instance, in respect of the sequential (Seq.) and integrated (Int.) approaches.

From these tables it can be verified that the proposed algorithm was able to reduce the number of crews in one unit in the instances VC52, VC260a and VC498. In the instances VC90 and VC639a, the number of crews increased in one unit; however, there was a significative reduction on the overtime payment

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

	Sequential			Integrated			Improvement	
Instance	Best	Avg.	Gap	Best	Avg.	Gap	Best	Avg
VC52	3936.96	3988.46	2.97	3873.31	3895.43	0.57	1.64	2.39
VC90	7519.20	7561.96	1.14	7476.57	7518.14	0.56	0.57	0.59
VC98a	9638.79	9675.39	1.16	9564.95	9594.67	0.31	0.77	0.84
VC98b	9500.16	9514.50	1.47	9376.67	9398.28	0.23	1.32	1.24
VC260a	28479.80	28714.00	1.43	28310.50	28329.11	0.07	0.60	1.36
VC260b	28064.80	28215.82	0.54	27921.40	27950.78	0.11	0.51	0.95
VC498	32673.50	32756.81	0.26	32500.10	32612.90	0.35	0.53	0.44
VC538	18917.00	18964.83	0.25	18887.20	18939.01	0.27	0.16	0.14
VC639a	39478.10	39586.00	0.27	39301.30	39398.46	0.25	0.45	0.48
VC639b	42557.60	42608.22	0.12	42336.20	42436.66	0.24	0.52	0.40
VC1036	48764.60	48840.86	0.16	48669.10	48778.18	0.22	0.20	0.13
VC1038	49223.90	49335.35	0.23	49085.40	49140.27	0.11	0.28	0.40

Table 4: Best schedules to instances VC52, VC90 and VC98a

Instance	VC52		VC90		VC98a	
Algorithm	Seq.	Int.	Seq.	Int.	Seq.	Int.
Number of vehicles	5	5	10	10	13	13
Vehicle waiting time (hh:mm)	12:57	12:57	16:13	16:13	11:58	11:58
Dead trip time (hh:mm)	03:50	03:50	08:24	08:24	15:20	15:20
Number of split blocks	0	0	4	4	7	7
Number of crews	12	11	20	21	27	27
Crew idleness (hh:mm)	11:31	08:31	18:44	14:28	17:05	13:29
Overtime payment (hh:mm)	02:25	05:52	14:13	10:00	06:38	05:09
Number of split duties	2	2	1	3	7	7
f_v	2640.4	2640.4	5168.4	5168.4	6809.8	6809.8
f_c	1296.6	1232.9	2350.8	2308.2	2829.0	2755.2
$f_v + f_c$	3937.0	3873.3	7519.2	7476.6	9638.8	9565.0

and on the crew idleness. For the other instances, o number of crews was the same in both algorithms. An interesting observation is that the integrated approach produced worse solutions to instances VC498 and VC538, but on the other hand the crew solution was better, resulting in a schedule with smaller cost. Also, the final solution produced by the integrated approach was better than the sequential one in all the instances.

5. 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.

Table 5: Best schedules to instances VC98b, VC260a and VC260b

Instance	VC98b		VC260a		VC260b	
Algorithm	Seq.	Int.	Seq.	Int.	Seq.	Int.
Number of vehicles	13	13	40	40	39	39
Vehicle waiting time (hh:mm)	10:21	10:21	19:15	19:15	13:22	22:02
Dead trip time (hh:mm)	14:34	14:34	37:54	37:54	37:54	37:54
Number of split blocks	6	6	22	22	23	23
Number of crews	25	25	76	75	76	76
Crew idleness (hh:mm)	12:50	09:29	91:31	83:03	84:10	79:07
Overtime payment (hh:mm)	15:13	10:59	07:14	08:46	11:28	07:19
Number of split duties	5	5	17	17	17	17
f_v	6744.1	6744.1	20189.0	20189.0	19785.3	19785.3
f_c	2756.1	2632.6	8290.8	8121.5	8279.5	8136.19
$f_v + f_c$	9500.2	9376.7	28479.8	28310.5	28064.8	27921.4

Table 6: Best schedules to instances VC498, VC538 and VC639a

Instance	VC498		VC538		VC639a	
Algorithm	Seq.	Int.	Seq.	Int.	Seq.	Int.
Number of vehicles	47	47	24	24	55	55
Vehicle waiting time (hh:mm)	37:38	39:27	65:26	65:28	50:36	51:45
Dead trip time (hh:mm)	23:36	24:48	18:59	20:23	52:03	52:37
Number of split blocks	25	26	3	3	25	25
Number of crews	94	93	53	53	109	110
Crew idleness (hh:mm)	55:03	44:51	50:55	47:57	57:47	51:29
Overtime payment (hh:mm)	24:36	22:17	33:17	29:48	30:10	17:48
Number of split duties	19	19	2	2	22	22
f_v	22851.1	22942.8	12755.6	12830.1	28133.0	28180.9
f_c	9822.4	9557.3	6161.4	6057.1	11345.1	11120.3
$f_v + f_c$	32673.5	32500.1	18917.0	18887.2	39478.1	39301.3

A comparison between the integrated approach and the traditional sequential one was 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 1.64% in relation to the best solution and up to 2.39% in relation to the average solution. In addition, it can be verified that the proposed algorithm is robust since the maximum variation of the final solutions is quite small, namely, 0.57%.

6. AKNOWLEGEMENTS

The authors thank CNPq (process 474831/2007-8), FAPEMIG (grant CEX 00357/09) and the Federal University of Ouro Preto for the support given to the development of this study.

References

Borndröfer, R., Grötschel, M., Pfetsch, M., 2006. Public transport to the fore! ORMS Today 33 (2), 30–40.

Table 7: Best schedules to instances VC639b, VC1036 and VC1038

Instance	VC639b		VC1036		VC1038	
Algorithm	Seq.	Int.	Seq.	Int.	Seq.	Int.
Number of vehicles	60	60	69	69	70	70
Vehicle waiting time (hh:mm)	50:42	52:16	66:07	69:13	67:41	69:10
Dead trip time (hh:mm)	32:59	56:06	50:28	51:41	50:32	51:46
Number of split blocks	25	27	35	35	32	33
Number of crews	116	116	136	136	137	137
Crew idleness (hh:mm)	59:44	49:23	67:57	60:41	78:26	67:10
Overtime payment (hh:mm)	31:06	22:01	43:20	37:16	30:50	26:25
Number of split duties	24	24	28	27	28	28
f_v	30525.4	30608.8	34552.2	34664.8	35027.3	35115.6
f_c	12032.2	11727.4	14212.4	14004.4	14196.6	13969.8
$f_v + f_c$	42557.6	42336.2	48764.6	48669.1	49223.9	49085.4

Bouzada, C. F., 2003. Costs of the Public Transport System by Bus (in portuguese). FUMEC, Brazil.

Darby-Dowman, K., Jachnik, J. K., Lewis, R. L., Mitra, G., 1988. Integrated decision support systems for urban transport scheduling: Discussion of implementation and experience. In: Daduna, J. R., Wren, A. (Eds.), Computer-Aided Transit Scheduling: Proceedings of the Fourth International Workshop. Springer Verlag, Berlin, pp. 226–239.

Freling, R., Huisman, D., Wagelmans, A., 2003. Models and algorithms for integration of vehicle and crew scheduling. Journal of Scheduling 6 (1), 63–85.

Freling, R., Wagelmans, A. P. M., ao, J. M. P. P., 1999. An overview of models and techniques for integrating vehicle and crew scheduling. In: Wilson, N. H. M. (Ed.), Computer-Aided Transit Scheduling. Springer Verlag, Berlin, pp. 441–460.

Freling, R. C., Boender, G. E., ao, J. M. P. P., 1995. An integrated approach to vehicle and crew scheduling. Tech. Rep. 9503/A, Econometric Institute, Erasmus University Rotterdam, Rotterdam.

Garey, M. R., Johnson, D. S., 1979. A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, New York.

Gendreau, M., Hertz, A., Laporte, G., 1994. A tabu search heuristic for the vehicle routing problem. Management science 40 (10), 1276–1290, linthicum.

Glover, F., Kockenberger, G., 2003. Handbook of Metaheuristics. Kluwer Academic Publishers.

Haase, K., Friberg, C., 1999. An exact branch and cut algorithm for the vehicle and crew scheduling problem. In: Wilson, N. H. M. (Ed.), Computer-Aided Transit Scheduling. Springer Verlag, Berlin, pp. 63–80.

Hansen, P., Mladenovic, N., 2001. Variable neighborhood search: Principles and applications. European Journal of Operational Research 130, 449–467.

Hansen, P., Mladenovic, N., Pérez, J. A. M., 2008. Variable neighborhood search. European Journal of Operational Research (191), 593–595.

Lourenço, H. R., Martin, O. C., Stützle, T., 2003. Iterated local search. In: Glover, F., Kochenberger, G. (Eds.), Handbook of Metaheuristics. Kluwer Academic Publishers, Boston.

Marinho, E. H., Ochi, L. S., Drummond, L. M. A., Souza, M. J. F., Silva, G. P., 2004. Tabu search applied to the bus driver scheduling problem (in portuguese). In: Proceedings of the XXXVI SBPO. São João Del Rey, Brazil, pp. 1471–1482.

Reis, J. V. A., 2008. Heuristics based on variable neighborhood search for the bus vehicle crew scheduling problem (in portuguese). Master's thesis, Escola Politécnica da USP, São Paulo, Brazil.

Schaerf, A., 1996. Tabu search techniques for large high-school timetabling problems. In: IEEE Transactions on Systems, Man, and Cybernetics. pp. 363–368.

Scott, D., 1985. A large linear programming approach to the public transport scheduling and cost model. In: Rousseau, J. M. (Ed.), Computer Scheduling of Public Transport 2. North Holland, Amsterdan, pp. 473–491.

Souza, M. J. F., Silva, G. P., oes, E. M. L. S., 2007. Vehicle scheduling: an heuristic approach (in portuguese). In: Transporte em transformação XI: trabalhos vencedores do prêmio CNT de Produção Acadêmica 2006. Positiva, Brasília, Ch. 2, pp. 39–57.

Stützle, T., 2006. Iterated local search for the quadratic assignment problem. European Journal of Operational Research 174, 1519–1539.