List Graph Colouring for Multiple Depot Vehicle Scheduling

Benoît Laurent[a,b]* and Jin-Kao Hao[b]

[a] Perinfo SA,
41 avenue Jean Jaurès, 67100 Strasbourg, France
Fax: +33 388449601 E-mail: blaurent@perinfo.com
[b] LERIA, Université d'Angers
2 Boulevard Lavoisier, 49045 Angers Cedex 01, France
E-mail: hao@info.univ-angers.fr
*Corresponding author

Abstract: This paper addresses a multiple depot vehicle scheduling problem (MDVSP) arising in public transportation. The general problem consists in assigning vehicles to trips while minimising the number of scheduled vehicles and the operational costs. The MDVSP considered here takes into account heterogeneous types of vehicles with complex relations among them. This special feature matches well situations encountered in practice, but makes the problem particularly difficult. We introduce a new formulation based on list graph colouring, from which an Iterative Tabu Search is developed for vehicle minimisation. The approach is assessed on 7 real-world benchmarks and yields highly satisfactory results in terms of solution quality and computation time.

Keywords: Multiple Depot Vehicle Scheduling; List Colouring; Tabu Search; Iterated Local Search.

Reference to this paper should be made as follows: Laurent B. and Hao J.-K. (xxxx) 'List Graph Colouring for Multiple Depot Vehicle Scheduling', *Int. J. of Mathematics in Operational Research*, Vol. x, No. x, pp.xxx–xxx.

Biographical Notes: Benoît Laurent is completing a PhD at the University of Angers (France) while working as an engineer in the R&D department of Perinfo SA. His research interests include issues related to vehicle and crew scheduling in public transport. He has published several papers in fully referred international journals and conference proceedings.

Dr. Jin-Kao Hao holds a full Professor position in the Computer Science Department of the University of Angers (France). His research lies in the design of effective heuristic and metaheuristic algorithms for solving large-scale combinatorial problems in various application areas such as telecommunication networks, transportation and bioinformatics. He has co-authored some ninety publications in international journals, book chapters and fully referred conference proceedings.

1 Introduction

The multiple depot multiple vehicle type scheduling problem can be defined as follows. We are given a set of trips and a set of vehicles housed in several depots of limited capacity. The fleet is heterogenous such that each vehicle belongs to a type: urban, articulated, scholarly, sub-urban, etc. In each depot, the available vehicles for each type are specified. A trip must be served by a certain type of vehicle. Upgrades may be allowed or not. The goal is to assign a vehicle to each trip such that the resulting schedule satisfies a set of constraints and minimises a cost function (e.g. the number of scheduled vehicles).

The MDVSP is a key step in the operational planning process of public transport companies. However, it is a challenging problem, shown to be NP-hard when two depots at least are considered Bertossi et al (1987). The complexity of the instances depends on several identified factors Kliewer et al (2006), such as:

- the number of timetabled trips,
- the number of depots,
- the integration or not of different types of vehicles.

Given its economic importance, the MDVSP has been studied for more than three decades now (Bodin et al (1983); Ribeiro and Soumis (1994); Forbes et al. (1994)). Because of the challenges evoked just before, it still arouses the community's interest. Until now, mathematical programming methods (Gintner et al (2005); Kliewer et al (2006); Hadjar et al (2006); Oukil et al (2007)) were mainly employed but metaheuristic algorithms gradually earn their respect (Pepin et al (2008); Laurent and Hao (2008)). The literature on the general MDVSP is surveyed in Section 3. We can already say that whatever their nature, exact or heuristic, many existing studies adopt simplifying hypotheses (e.g. homogeneous fleet) which do not match the typical real situations. Indeed, considering a heterogeneous fleet in the MDVSP is a challenging issue. In particular, it leads to an explosive size increase for models that consider explicit connections between trips. In the case of metaheuristic methods, even generating an initial solution remains a difficult task.

The real-world application that motivated our work combines some practical, yet difficult features. First, the number of depots is very high (> 60). Second, we must integrate many categories (> 8) of vehicles (heterogeneous fleet) with special relations between them. Finally, the number of trips to be planned is large (at least several hundreds).

To deal with this challenging problem, we introduce a new formulation of the multiple depot multiple vehicle type scheduling problem as a list-graph colouring problem, denoted $\mathcal{L}$-GCP hereafter. Since graph colouring problems have been extensively studied, we benefit from an appealing framework that comprises theoretical results and effective algorithms. Based on the formulation, we propose a solution procedure using Iterated Tabu Search able to rapidly generate good quality solutions with respect to the number of vehicles.

The remainder of this article is structured as follows. After a description of the MDVSP, we review the main works conducted on this problem in Section 3. In Section 4, we present a formulation of the MDVSP as a list-graph colouring problem. Then, we describe the main components of our Iterated Tabu Search

algorithm. Section 6 is dedicated to computational experiments. The last section discusses several aspects related to the real tests that were conducted.

2 Multiple Depot Vehicle Scheduling Problem

In this section, we present the MDVSP, which has the following input data:

- *Trips*: a set of n commercial trips, each one being characterized by an origin and a destination with associated starting and ending times; each trip needs to be covered by exactly one vehicle belonging to the required type (see next points). Two trips are said *compatible* if after serving the first one, a vehicle has enough time to reach the beginning of the second one. Without loss of generality, we assume that the trips are ordered by increasing starting time.
- *Vehicles and depots*: a fleet of p vehicles housed in m depots ($1 < m \ll n$) of limited capacity. Each vehicle will be assigned a duty, i.e. a sequence of consecutively compatible trips.
- *Categories*: a vehicle belongs to a given category (or type) regarding its equipments, degree of comfort and number of seats. There exists relations between categories that govern the possible substitution of a vehicle by another one to operate a trip. We assume a common speed for all vehicles, i.e. travel times do not depend on the category.
- *Trips without passengers*: before or after a commercial trip, a vehicle may perform a connection without passengers. The trip leaving a depot to reach the first trip of the duty is called a pull-out trip. Its symmetrical counterpart is called a pull-in trip. The connections between consecutive trips are called deadhead trips.

A valid vehicle schedule must satisfy the following five constraints:

1. **Complete cover constraints**: all the trips must be covered by exactly one vehicle.
2. **Category constraints**: each trip must be handled by a vehicle of the required type. Substitutions are possible if they satisfy the relations among vehicle categories.
3. **Feasible sequence constraints**: two consecutive trips covered by a vehicle must be compatible.
4. **Depot attachment constraints**: the vehicles return to the depot they start from.
5. **Depot capacity constraints**: the number of vehicles used in each depot does not exceed the depot capacity.

Moreover, a valid vehicle schedule needs to minimize an objective function, composed of fixed and operational costs.

1. **Fixed cost related to scheduled vehicles**: each scheduled vehicle represents a fixed and important cost.
2. **Operational costs**: these costs are induced by non-commercial trips, i.e. pull-in, pull-out and deadhead trips.

The multiple depot multiple type vehicle scheduling problem consists in determining a vehicle schedule that satisfies the set of imperative constraints 1)-5) while minimizing the objective function. The cost structure reflects the major importance accorded to the minimisation of the number of vehicles compared to the operational costs. Consequently, we restrict ourselves to the first objective in this paper.

3 Related Works

The literature on the MDVSP offers a range of solution methods developed in the last three decades. These methods can be classified in two main categories: those which assume a homogeneous fleet and those which consider different types of vehicles.

3.1 Homogeneous Fleet

The early works on this problem focus on heuristic algorithms (see for instance Bodin et al (1983); Carraresi and Gallo (1984); Rousseau et al (1988)). Two main approaches coexist essentially. The first one consists in clustering the trips and assigning them to the depot first, and then scheduling the vehicles in each depot separately. In the second approach, the whole fleet is first scheduled as if there were only one depot and then the resulting schedules are assigned to each depot. This latter idea is still employed in recent studies (Pepin et al (2008); Laurent and Hao (2008)).

Since the end of the 80's, several exact algorithms have been proposed. The models employed belong to one of the three following categories:

- single-commodity flow formulations (e.g. Carpenato et al. (1989); Fischetti et al (2001)),
- multicommodity flow formulations (e.g. Forbes et al. (1994); Löbel (1997)),
- set partitioning formulations (e.g. Ribeiro and Soumis (1994); Hadjar et al (2006); Oukil et al (2007)).

A detailed description of these models is beyond the scope of this paper. We refer interested readers to Desaulniers and Hickman (2007) and Ceder (2007) for a general presentation of transportation issues, including vehicle scheduling problems. For a dedicated and up-to-date survey on vehicle scheduling models, the reader is referred to Bunte et al (2006).

Integer linear programming remains undoubtedly the most popular approach to the MDVSP. Metaheuristic algorithms appeared very recently: Large Neighbourhood Search (LNS) and Tabu Search (TS) in Pepin et al (2008) and Iterated Local Search (ILS) in Laurent and Hao (2008). These two articles reported comparisons

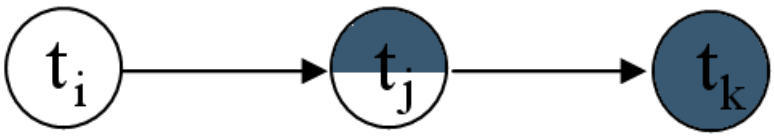


Figure 1 Unfeasible block

between different heuristics including these metaheuristic algorithms, a heuristically applied CPLEX MIP solver, a Lagrangian heuristic and a column generation heuristic. Both studies showed a dominance of the column generation heuristic in terms of solution quality. Nevertheless, when a compromise between quality and computation time is aimed, metaheuristics become a very competitive alternative.

3.2 Heterogeneous Fleet

Most of the foregoing papers assume a homogeneous fleet such that trips can be performed by any vehicle. On the one hand, this assumption does not hold in practical situations. On the other hand, allowing a heterogeneous fleet increases the problem complexity. As already identified in Löbel (1998), obstacles may arise in the construction of initial solutions by means of early heuristics. Suppose for example that we are given three trips, t_i, t_j and t_k and two vehicles v and v'. We shall assume that t_i can only be performed by v, t_k only by v', while no restriction is imposed upon t_j. A schedule first - cluster second approach reduces the multiple depot formulation to a single-depot relaxation. Since it only considers feasible links between trips, it could produce the infeasible solution depicted in Figure 1. Splitting such infeasible blocks into feasible parts can lead to suboptimal solutions.

The book of Ceder (2007) contains a heuristic proposal to cope with different types of vehicles. However, it assumes the existence of a total order relation between types, which is not always the case in practice (including this study). In general, the consideration of heterogeneous types of vehicles greatly increases the complexity of the trip-connection-based models. As a consequence, an approach commonly observed aims at heuristically reducing the number of connections. In Haghani and Banihashemi (2002) for instance, an assumption is made that no evening trip will be served directly after a morning trip, resulting in a reduction of the model size of up to 40% without sacrificing optimality. Another response to the explosive increase of the model size avoids the explicit consideration of all the connections (see Kliewer et al (2006)). The proposed modeling approach relies on a time-space based network. Coupled with a fix-and-optimize heuristic in Gintner et al (2005), this approach is able to solve real-world problem instances with up to thousands of trips and twelve vehicle types. In a nutshell, a series of subproblems is first solved to identify stable chains of trips, i.e. trips that fit well together. These stable chains are subsequently aggregated to form a single trip in the general problem. It should be noted that these instances possess a special structure. The time-space network is especially relevant when the number of locations (stations) involved in the problem is low compared to the number of trips. Unfortunately, this is not the case in our encountered inter-city scenarios.

4 Formulation as a List Graph Colouring Problem

In this section, we formulate the MDVSP as a list graph colouring problem ($\mathcal{L}$-CGP). This formulation leaves aside the secondary objectives to focus on the number of scheduled vehicles. Beforehand, we introduce some needed notations.

4.1 Notations

- n, p: the number of trips and vehicles.
- $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$: the set of n trips, ordered by increasing starting times.
- (s_i, e_i): the starting and ending times of a trip $t_i \in \mathcal{T}$.
- τ_{ij}, $i, j \in \{1...n\}$, $i \neq j$: the transfer time from the end of trip t_i to the start of trip t_j.
- $\mathcal{V} = \{1, 2..., p\}$: the fleet of p vehicles.
- $\mathbb{L}_i \subset \mathcal{V}$: the set of vehicles that can handle trip $t_i \in \mathcal{T}$.

The depots do not appear explicitly here. Actually, their role lies in the definition of the available fleet $\mathcal{V}$, done with regards to the capacities.

4.2 List Graph Colouring Problem

Generally speaking, a list graph colouring problem is an extension of the well-known graph colouring problem (GCP). It arises when each vertex has associated specifications on the colours that are admissible Tuza (1997). This problem is also designated in the literature under the term *restricted graph colouring problem* De Werra (1997).

Formally, given an undirected graph $G = (V, E)$ where $V = \{1, \ldots, n\}$ and $E \subset V^2$ are the sets of vertices and edges respectively, a set of colours Γ, and a list of sets of colours $\mathcal{L} = (L_1, \ldots, L_n)$ with $L_i \subset \Gamma$, $1 \leq i \leq n$, we seek for a vertex colouring ϕ such that:

- $\phi(i) \in L_i, \forall\, i \in \{1, \ldots, n\}$ and
- $\phi(i) \neq \phi(j), \forall\, (i, j) \in E$.

The $\mathcal{L}$-GCP is NP-complete in the general case, since it reduces to the k-colouring problem if $L_i = \Gamma$ for each $i \in \{1, \ldots, n\}$.

In order to model MDVSP, we can define an undirected incompatibility graph $G = (V, E)$ such that:

- the vertex set V matches the trips in $\mathcal{T}$,
- the pre-specified set of colours L_i for each vertex i corresponds to the vehicles that can handle trip t_i, i.e. $L_i = \mathbb{L}_i$,

- a pair of vertices $(i, j) \in V^2, i < j$ are joined by an edge in E, if $e_i + \tau_{ij} > s_j$ or $\mathbb{L}_i \cap \mathbb{L}_j = \emptyset$. In other words, two trips t_i, t_j are incompatible if there is no sufficient time for a vehicle to reach the start location of t_j when t_i is achieved. Furthermore, t_i and t_j are necessarily incompatible if they have no vehicle in common in their list.

Notice that the proposed translation of the MDVSP into a list graph colouring problem captures a decision problem that can be stated as follows: given an available fleet of vehicles (= the set of colours Γ), one aims to find a schedule (= a colouring) that satisfies all the constraints with at most these colours.

To deal with the optimisation version of the original problem (i.e. vehicle minimisation), we introduce an upper bound UB on the maximal number of vehicles (= the number of colours) that can be used, thus defining the $UB - \mathcal{L}$-GCP. The management of UB is discussed in Section 5.1. An alternative solution would have been the elimination of a colour each time a proper list-colouring is found. Such an approach is commonly used for solving the conventional GCP, however, we did not retain this idea since the choice of the colour to be removed is a problem itself (see Section 6.4 for more details).

Figure 2 illustrates the transformation of a small MDVSP, implying 4 trips and 3 vehicles, into a $\mathcal{L}$-GCP. The original problem is represented by a directed graph where the vertices labeled $\{1, 2, 3\}$ are the vehicles and trips are labeled by $\{t_1, t_2, t_3, t_4\}$. Two trips joined by an arc are compatible, whereas a vehicle can perform a trip if and only if there is an arc between them. The figure shows the undirected graph resulting from the translation. Two possible solutions are finally displayed depending on the value of UB.

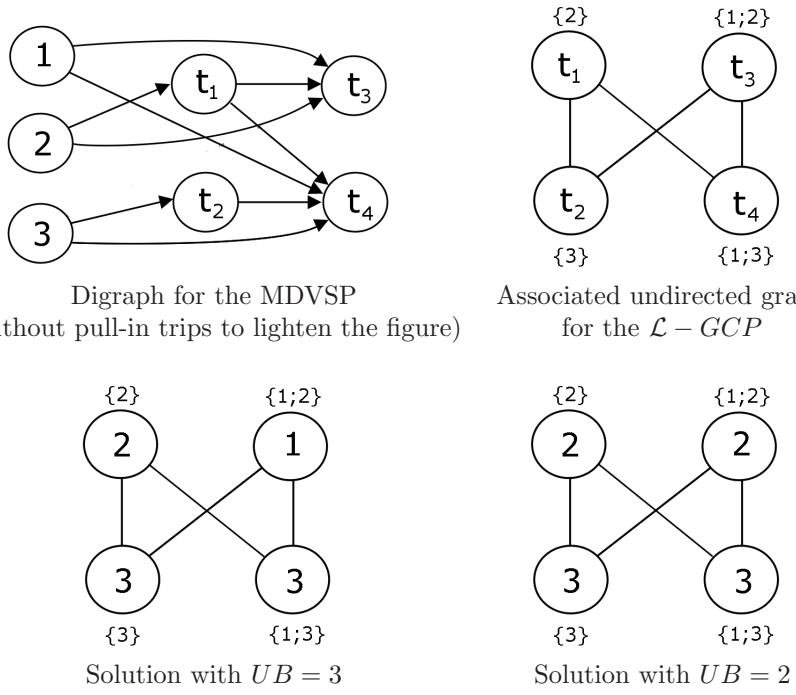
5 Solution Approach

In this section, we propose an Iterated Tabu Search (ITS) algorithm dedicated to the $\mathcal{L}$-GCP. Contrary to many approaches that consider complete but often infeasible configurations, our algorithm evolves within a search space of partial but feasible colourings.

5.1 Iterated Tabu Search Algorithm

The general idea of our algorithm consists in solving a series of list colouring problems with a fixed number UB of colours.

Our ITS algorithm follows the scheme of Iterated Local Search (see Lourenço et al (2002)). It begins with an initial solution and then alternatively repeats an intensification phase and a diversification phase. The intensification phase, assured by a Tabu Search procedure Glover and Laguna (1997), improves the current solution s by iteratively replacing s by a neighbouring solution s' taken from a given neighbourhood relation. When η iterations without strict improvement have elapsed, the diversification phase is triggered with the goal of leading the search to a new promising area of the search space. Diversification is realized here by randomly perturbing the best solution found during the previous intensification phase.



Digraph for the MDVSP (without pull-in trips to lighten the figure)

Associated undirected graph for the $\mathcal{L} - GCP$

Solution with $UB = 3$

Solution with $UB = 2$

Figure 2 Transformation of the MDVSP into a $\mathcal{L} - GCP$

An acceptation test is finally carried out to decide whether the so-far best solution should be updated.

The management of UB is performed at the highest level of ITS, the role of the Tabu Search procedure being the resolution of the $UB - \mathcal{L}$-GCP. More precisely, the functioning of ITS is the following:

1. Initially, set UB to the number of available vehicles: $UB = p$.

2. Seek a UB-list-colouring using TS, i.e. find a schedule with the UB given vehicles.

3. If a proper UB-list-colouring is found at Step 2), $UB = UB - 1$. Otherwise, UB remains unchanged.

4. Apply a diversification mechanism (see Section 5.6): basically a subset of colours is chosen, and all vertices receiving these colours are uncoloured. Goto Step 2.

5. Once a stopping criterion is met (see Section 5.7), report the best solution found so far.

We detail hereafter the elementary components of the ITS algorithm: search space, neighbourhood, evaluation function, perturbation and acceptation criterion.

5.2 Search Space and Initial Configuration

A configuration σ in the *unconstrained* search space Σ represents a (possibly partial) assignment of colours to vertices: $\sigma : V \rightarrow \{1, \ldots, p\} \cup \{\epsilon\}$, where ϵ indicates an uncoloured vertex. If a configuration σ contains at least one ϵ value, then it is a partial colouring; otherwise, it is said complete. Associated to σ, we define the set of non-coloured vertices: $V_{uncl} = \{i \in V \mid \sigma(i) = \epsilon\}$.

The search is confined within the subspace $\Omega \subset \Sigma$ of feasible colourings:

$$\Omega = \{\sigma \in \Sigma \mid (\sigma(i) \neq \sigma(j)) \ \vee \ (\sigma(i) = \epsilon) \ \vee \ (\sigma(j) = \epsilon), \ \forall (i,j) \in E \quad \text{and} \\ (\sigma(i) \ \in \ \mathbb{L}_i \ \vee \ (\sigma(i) = \epsilon), \ \forall i \in \{1, \ldots, n\}\}$$

It is clear that Ω contains both the complete and partial colourings. The word *solution* is used to designate any complete colouring of Ω. Note that the idea of using partial configurations was also recently reported in other studies (see for instance, Vasquez (2002), Blöchliger and Zufferey (2008)).

The empty colouring has no coloured vertex and clearly constitutes a trivial feasible configuration. Our ITS algorithm uses this colouring as its initial configuration. In other words, no vehicle is assigned at the beginning of the search. The rest of the algorithm constantly maintains the exploration within the subspace Ω.

5.3 Evaluation Function

Like any local search algorithm, our ITS algorithm needs an evaluation function to estimate the quality of the configurations visited during the search. Along with the neighbourhood structure, the evaluation function is of paramount importance for the performance of our ITS algorithm.

Starting from an empty colouring, ITS aims at finding a feasible *complete* colouring. For this purpose, ITS dynamically colours and uncolours vertices until a complete and feasible colouring is found. So for a given configuration σ, one can evaluate the quality of σ by the number of uncoloured vertices:

$$f_1(\sigma) = |V_{uncl}|$$

Intuitively, this evaluation function may not be informative enough; it simply cannot distinguish many equally valued configurations. Indeed, this function has only $|V|$ different values while the number of configurations in the search space Ω is much larger.

As an alternative, we devise a second evaluation function:

$$f_2(\sigma) = \sum_{i \in V_{uncl}} (w \times |\mathbb{L}_i| + \delta(i))$$

where w is a parameter (set to 10000 in our experiments) and $\delta(i)$ is the degree of vertex i. f_2 tends to favor the colouring of vertices of smaller domains (ties being broken according to the degree).

The relative pertinence of both functions is evaluated in Section 6.2.

5.4 Neighbourhood Structures

The neighbourhood used by ITS is a simple form of ejection chains. The algorithm also embeds a candidate list $\mathcal{L}_{cand} \subset V_{uncl}$ containing the most promising vertices, i.e. a subset of the non-coloured vertices, the domain of which is small (ties being again broken according to the degree). The idea underlying the candidate list is to allow the search to focus on critical movements (see Section 6.2). Its size is parameterized by α:

$$|\mathcal{L}_{Cand}| = \min(\alpha \times |V|, |V_{uncl}|)$$

.

The mechanism designed to generate a neighbour σ' of the current configuration σ follows the scheme sketched in Algorithm 1. A non-coloured vertex belonging to the candidate list $i \in \mathcal{L}_{Cand}$ is selected. If the number of colours already used $nb_{cl}(\sigma)$ in σ is strictly inferior to the upper bound UB, a colour is randomly chosen from the list $\mathbb{L}_i$. Otherwise, the colour is selected from $\mathbb{L}_i \cap U_{cl}(\sigma)$ where $U_{cl}(\sigma)$ is the set of colours in use[a]. The assignment of cl to i becomes effective even if it leads to conflicts. In case some conflicts occur, we try to assign to each conflicting vertex j a new colour taken from $\mathbb{L}_j \cap U_{cl}(\sigma)$. If the attempt fails, j is uncoloured and inserted in V_{uncl}. $\mathcal{L}_{Cand}$ is updated if needed.

Algorithm 1 Pseudo-code for the generation of neighbouring configurations

Require: a configuration σ, the set of used colours $U_{cl}(\sigma)$

```
i ← rand(L_Cand)
if (nb_cl(σ) < UB) then
  cl ← rand(U_cl(σ))
else
  cl ← rand(L_i ∩ U_cl(σ))
end if
σ(i) ← cl

{Repair attempt}
for all j such that σ(j) = cl do
  if ((i, j) ∈ E) then
    cl' ← repair_attempt(j, nb_cl(σ), UB)
    if cl' ≠ ∅ then
      j ← cl'
    else
      j ← ϵ
      V_uncl ← V_uncl ∪ {j}
      update(L_Cand)
    end if
  end if
end for
```

[a] If $\mathbb{L}_i \cap U_{cl}(\sigma) = \emptyset$, a new vertex is selected. When no vertex can be coloured, the Tabu Search phase is halted. We do not mention these cases in the algorithm for the sake of simplicity.

5.5 Tabu List and Tabu Tenure

The name of the Tabu Search method refers to the short term memory that keeps track of the attributes of configurations previously visited or of the movements recently applied. Here, a movement is composed of a set of (vertex, colour) pairs. It is considered as tabu if the exact corresponding set appears in the tabu list.

The tabu tenure indicates the number of iterations during which a movement is proscribed. In our case, the tabu tenure varies from β to 2β where β is a user parameter (see Section 6.2). Once the tenure has elapsed, the movement is again accessible. Note that we also use an aspiration criterion that overrules the tabu restriction of a movement when it improves the best configuration encountered so far.

5.6 Perturbation Mechanisms and Acceptance Criterion

In order to escape from local optima and to explore new regions of the search space, ITS applies perturbations to the best local minimum. Our perturbation mechanism randomly destroys a fraction γ of the best configuration, subsequently rebuilt during the Tabu Search phase. Despite its apparent simplicity, this perturbation furnishes satisfactory results.

The acceptance criterion plays a role in the balance between intensification and diversification. Our acceptance criterion clearly privileges intensification since the best solution resulting from a Tabu Search phase is accepted if and only if it improves the best overall solution.

5.7 Stopping Criterions

Our ITS algorithm is halted when one of the two criteria is met:

- the maximal allowed time has elapsed (set to $n/10$ seconds),
- a lower bound is reached.

Before actually running the algorithm, we solve a max-clique problem on the relaxed $\mathcal{L}$-GCP, i.e. without taking into account the list of allowed colours. For this purpose, we implemented an efficient algorithm given in Pullan and Hoos (2006). The size of the maximum clique $\omega(G)$ constitutes a lower bound for the chromatic number $\chi(G)$ of the GCP. This number is also a lower bound for the number of colours of the $\mathcal{L}$-GCP.

6 Experimental Results

6.1 Benchmarks

The experiments that we carried out rely on 7 instances[b] coming from a real-world situation for inter-city transport. To our knowledge, the literature does not

[b]These instances are defined using the DIMACS format and available upon request

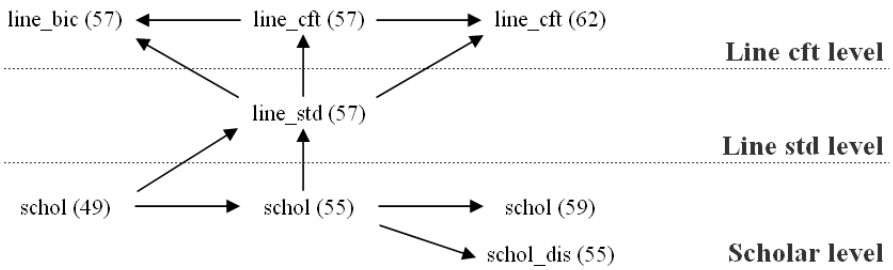


Figure 3 Relations between categories of vehicles (transitivities are not represented)

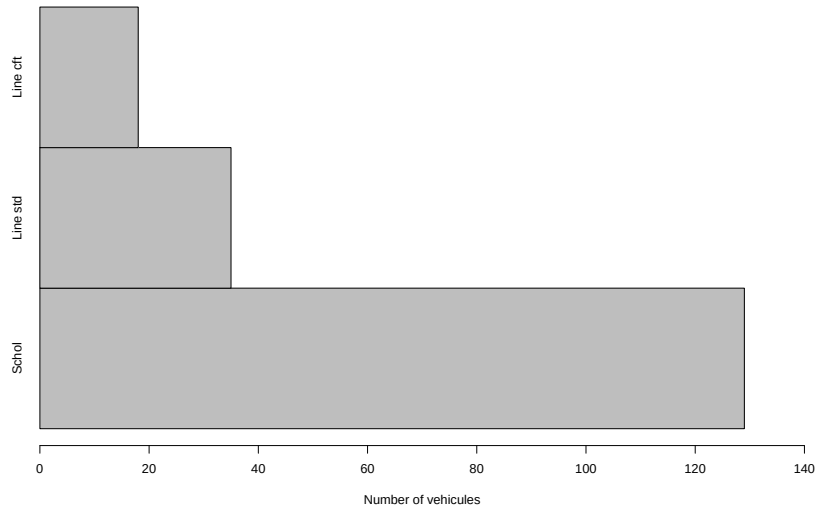


Figure 4 Distribution of the vehicles among the different levels

offer freely available benchmarks integrating different types of vehicles. The problem instances all contain 683 vertices (or trips). The number of initially available colours (vehicles) amounts to 182.

The vehicles belong to 8 categories grouped into 3 levels. There exists complex relations among these categories which are depicted in Figure 3. The numbers between brackets indicate the available seats in buses. The vertical arrows imply allowed upgrades between categories while horizontal ones model possible substitutions within one level. In this case, the restrictions are due to a lack of seats or equipments. For example, *schol* and *schol_dis* both belong to the same level. The latter category contains buses specially equipped for disabled passengers. Therefore, a trip requiring a *schol* bus can be *a fortiori* handled by a bus of *schol_dis* category. The opposite cannot be operated. As shown in Figure 4, the numbers of available vehicles in each level form a pyramid: the higher the level is, the smaller the number of vehicles is.

The instances *full_upgrades.col*, *1_upgrade.col* and *no_upgrade.col* correspond to the respective cases where all upgrades, one level of upgrade and no upgrades are allowed. The incidences on the degrees of the vertices and their domain size can be viewed in Table 1. As expected, the more constrained the problem is, the smaller the domains and the higher the degrees are.

The travel times between pairs of locations have been pre-computed by means of a Geographical Information System (GIS). However, we wished to examine the sensitivity of the problem to speed variations. In the instances *full_upgrades_0_8.col*, *full_upgrades_0_9.col*, *full_upgrades_1_1.col* and *full_upgrades_1_2.col*, we multiplied these travel times by a factor of 0.8, 0.9, 1.1 and 1.2 respectively.

Name	Density	Vehicles/Trip	
	(%)	Avg	Sd
no_upgrade.col	56.6	85.58	47.08
1_upgrade.col	42.3	109.34	62.66
full_upgrades.col	29.8	121.68	69.18
full_upgrades_0_8.col	28.3	—	—
full_upgrades_0_9.col	29.1	—	—
full_upgrades_1_1.col	30.3	—	—
full_upgrades_1_2.col	30.9	—	—

Table 1 Characteristics of the instances

In terms of size, these instances are comparable to those mentioned in previous studies (see Section 3). They correspond to a typical extra-urban situation in France: some 62 depots are spread over the considered geographic area. Most of them are actually car-parks or domiciles of drivers. This situation can be seen on Figure 5: on the distribution of the vehicles among depots, a large number of them have less than 5 vehicles. Because of the considered numbers of depots and categories, these instances may cause great difficulties for classical trip-connection-based models.

6.2 Settings and Parameters

Our ITS algorithm was coded in C++, compiled with VC++ 8.0 (flag -O3), on a laptop equipped with a 2 Ghz T7200 Intel Core and 2Gb RAM running Windows XP. Due to the stochastic nature of the algorithm, 10 runs on each instance were performed.

In order to conceive the best performing algorithm, we applied a 2-level full factorial experiment (see the book of Montgomery (2004) for a survey on the design of experiments) to determine the influence of the following components:

- the pertinence or not of using a neighbourhood with repair attempts,
- the decision to employ the evaluation function $f_1(\sigma) = |V_{uncl}|$ or $f_2(\sigma) = \sum_{i \in V_{uncl}} w \times |\mathbb{L}_i| + \delta(i)$,
- the ratio of destruction γ for the perturbation moves,

From this experiment, the type of neighbourhood reveals to be a significant factor. A Student t-test rejected the null-hypothesis at a 0.1 threshold. The evaluation function and the ratio of destruction seem less significant. We do not retain

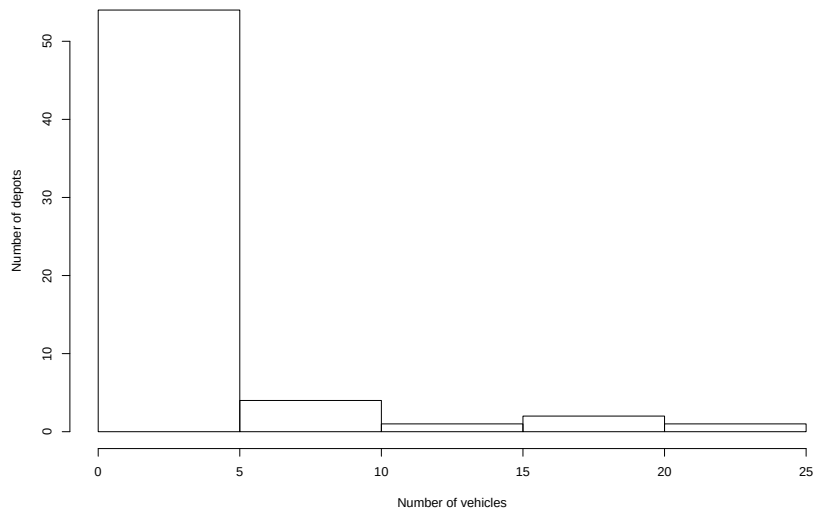


Figure 5 Distribution of the vehicles among the different depots

f_2 in the algorithm since its use appears counterproductive: it prevents the search from attaining the best known solution for the instance *full_upgrades_0_8.col* (134 colours instead of 133). A possible explanation is that f_2 does not sufficiently maintain randomness in the algorithm. Furthermore, it may be redundant with the candidate list. For a definite conclusion about its relevance, some additional data and experiments would be needed. The use of a partial destruction ($\gamma = 0.5$) slightly improves the results even if it is not significant statistically speaking.

In our ITS algorithm, three parameters need to be tuned:

- the size of the candidate list α,
- the size of the tabu list β,
- the number of non-improving moves η before halting the tabu search.

The size of the candidate list clearly influences the computation time. Figure 6 represents the average time needed to attain the best solution. From this figure, one observes that a strong intensification accelerates the search. As a consequence, α was assigned the value 0.01.

The last two parameters, β and η were set to $|\mathcal{L}_{cand}|$ and 100 respectively.

6.3 Results

Table 2 gathers the results of the experiments carried out on the 7 instances and the obtained results. As mentioned in Section 5.7, we first solved a max-clique problem to get a lower bound on the required number of colours (see column *Lower Bound*). In order to check the validity of this lower bound, we solved a relaxed

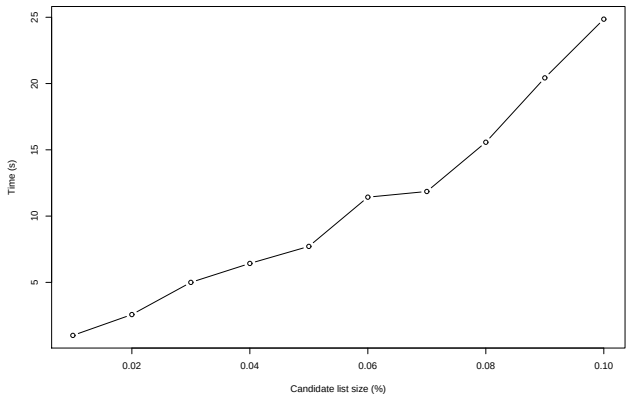


Figure 6 Influence of parameter α on the computation time

problem in parallel, i.e. without taking into account the resources limitations (*Unlimited Resources*). The last four columns of this table deal with the Best value (*Min*), the worst value (*Max*), the average value (*Avg*) and the average time in seconds (*Time (s)*) to find the best solution.

Name	Lower Bound	Unlimited Resources	Min	Max	Avg	Time (s)
no_upgrade.col	159	159	159	159	159.0	0.6
1_upgrade.col	135	135	135	135	135.0	7.3
full_upgrades.col	133	133	135	135	135.0	4.7
full_upgrades_0_8.col	129	129	133	134	133.4	5.4
full_upgrades_0_9.col	132	132	135	135	135.0	2.3
full_upgrades_1_1.col	134	134	137	138	137.1	7.0
full_upgrades_1_2.col	138	138	141	141	141	4.0

Table 2 Results of the ITS algorithm in terms of quality and response time

We can make several comments from this table. First, the size of the maximum clique $\omega(G)$ is always equal to $\chi(G)$ the chromatic number of the relaxed $\mathcal{L}$-GCP. This demonstrates the validity of the lower bound. It also means that the graphs encountered are 1-perfect. This feature is consistent with observations reported by other authors about real-world graphs Couder (1997).

The second comment concerns the quality of the solutions. For the instances *no_upgrade.col* and *1_upgrade.col*, an optimal solution is reached. For the remaining cases, the gap with the lower bound never exceeds 4 % in the worst case. Even if it is not provable, we can reasonably assume that these gaps are due to the limitations on the available vehicles. The algorithm appears robust on these instances since the average value always reaches the best value in 5 cases over 7. As expected, the different scenarios in travel speed cause minor alterations in densities (see Table 1) but significant increases in the number of vehicles. Finally, despite the size of the instances, the computation time remains very short.

6.4 Discussions

In this section, we first discuss a path that was also explored for solving the $\mathcal{L}$-GCP: we attempted to further transform the problem into a classical GCP. We also give more details about the real data we used for our experimentations. Finally, we consider an over-constrained scenario which may arise if a lack of resources occurs.

When considering its decision version, the $\mathcal{L}$-GCP can be reduced in two steps into a classical GCP. Proposed in Biró et al (1992), the reduction first transforms the $\mathcal{L}$-GCP into a precolouring extension problem (i.e. a GCP on an already partially coloured graph), which is subsequently reduced to a GCP. Since many effective algorithms are available for the GCP, such a reduction seems appealing at first sight, but is actually not applicable in our case. A classical approach to handle the optimisation version of a colouring problem consists in solving a series of decision problems with a decreasing number of colours. For the $\mathcal{L}$-GCP however, the choice of the colour (= vehicle) to be removed at each iteration is a problem itself since each vehicles has a different status.

The original data actually contain 921 trips, 235 vehicles and 15 categories. However, the set of trips can be split to form several subproblems completely insulated from each other. The case study corresponds to a mixed urban/extra-urban situation: some vehicles are especially dedicated to urban areas while others only deal with extra-urban trips. We only extracted the largest subproblem to simplify the presentation but the algorithm can address the whole problem at once.

In these benchmarks, the number of available vehicles exceeds the needs (182 available vehicles against 159 used in the worst case). The explanation is the following: every day, additional punctual trips must be inserted in the regular schedule. If an over-constrained situation happens, i.e. the vehicles resources are not sufficient to face the workload, the model presented in this paper would be appropriate since it natively manages a list of uncoloured vertices.

7 Conclusion

The Multiple Depot Vehicle Scheduling Problem is of paramount importance in the operational planning process of public transport systems. Despite the abundant literature on this topic, many papers adopt simplifying hypothesis, rendering the methods hardly applicable in real situations.

In this paper, we dealt with a real-world MDSVP with heterogeneous fleet of vehicles and complex relations among vehicle categories. We introduced a new model for the MDVSP using list graph colouring with the goal of minimising the scheduled vehicles. Associated to this model, we proposed an Iterated Tabu Search integrating some original features: an efficient neighbourhood scheme, a dynamic candidate list strategy and a simple but effective destruction perturbation mechanism.

We assessed the practical effectiveness of the algorithm on 7 real-world benchmarks. The experiments showed its capacity in yielding fast and high quality solutions regarding the number of vehicles. Thanks to these interesting properties, the algorithm can be used by various metaheuristics to generate good initial solutions. Finally, let us mention that the algorithm described in this work is part of a real

vehicle scheduling system that handles both fixed costs related to the scheduled vehicles and operational costs.

Acknowledgments: This work was partially supported by the French Ministry for Research and Education through a CIFRE contract (number 176/2004). Finally, we would like to thank Valérie Guihaire for various constructive discussions. The reviewers of the paper are greatly acknowledged for their helpful comments.

References

Bertossi, A., Carraresi, P. and Gallo G. (1987) 'On some matching problems arising in vehicle scheduling models', *Networks*, Vol. 17, No. 1, pp.271–281.

Blöchliger, I. and Zufferey, N. (2008) 'A graph coloring heuristic using partial solutions and a reactive tabu scheme', *Computers & Operations Research*, Vol. 35, No. 3, pp.960–975.

Biró, M., Hujter, M. and Tuza, Zs. (1992) 'Precoloring Extension. I: Interval Graphs', *Discrete Mathematics*, Vol. 100, No. 1-3, pp.267–279.

Bodin, L., Golden, A. and Ball, M. (1983) 'Routing and scheduling of vehicles and crews: The state of the art', *Computers & Operations Research*, Vol. 10, No. 2, pp.63–211.

Bunte, S., Kliewer, N. and Suhl,L. (2006) 'An overview on vehicle scheduling models in public transport', *Computer-Aided Scheduling of Public Transport*, Leeds UK, Springer Verlag.

Carraresi, P. and Gallo, G. (1984) 'A multi-level bottleneck assignment approach to the bus drivers rostering problem', *European Journal of Operational Research*, Vol. 16, No. 2, pp.163–173.

Carpenato, G., Dell'Amico, M., Fischetti, M. and Toth, P. (1989) 'A branch and bound algorithm for the multiple depot vehicle scheduling problem', *Networks*, Vol. 19, No. 5, pp.531–548.

Ceder, A. (2007) 'Public Transit Planning and Operation: Theory, Modeling and Practice', *Elsevier*, Butterworth-Heinemann, 640 p., Oxford, UK.

Couder, O. (1997) 'Exact coloring of real-life graphs is easy', *Design Automation Conference*, pp.121–126.

Desaulniers, G. and Hickman, M. (2007) 'Public transit', *Handbooks in Operation Research and Management Science*, pp.69–120.

De Werra, D. (1997) 'Restricted coloring models for timetabling', *Discrete Mathematics*, Vol. 165/166, pp.161-170.

Fischetti, M., Lodi, A., Martello, S. and Toth, P. (2001) 'A polyhedral approach to simplified crew and vehicle scheduling problems', *Management Science*, Vol. 27, No. 6, pp.1–18.

Forbes, M., Holt, J., and Watts, A. (1994) 'An exact algorithm for multiple depot bus scheduling', *European Journal of Operational Research*, Vol. 72, No. 1, pp.115–124.

Gintner, V., Kliewer, N. and Suhl, L. (2005) 'Solving large multiple-depot multiple-vehicle-type bus scheduling problems in practice', *OR Spectrum*, Vol. 27, No. 4, pp.507–523.

Glover, F. and Laguna, M. (1997) 'Tabu search', *Kluwer Academic Publishers*, Boston.

Hadjar, A., Marcotte, O., and Soumis, F. (2006) 'A branch-and-cut algorithm for the multiple depot vehicle scheduling problem', *Operations Research*, Vol. 54, No. 1, pp.130–149.

Haghani, A. and Banihashemi, M. (2002) 'Heuristic approaches for solving large-scale bus transit vehicle scheduling problem with route time constraints', *Transportation Research Part A: Policy and Practice*, Vol. 36, No. 4, pp.309–333.

Kliewer, N., Mellouli, T. and Suhl, L. (2006) 'A timespace network based exact optimization model for multi-depot bus scheduling', *European Journal of Operational Research*, Vol. 175, No. 3, pp.80–87.

Laurent, B. and Hao, J.-K. (2008) 'Iterated local search for the multiple vehicle scheduling problem', *Computers & Industrial Engineering (to appear).*

Löbel, A. (1997) 'Optimal Vehicle Scheduling in Public Transit', *PhD thesis*, Technische Universität, Berlin.

Löbel, A. (1998) 'Vehicle scheduling in public transit and lagrangian pricing', *Management Science*, Vol. 44, No. 12, pp.1637–1649.

Lourenço, H., Martin, O., and Stützle, T.(2002) 'Iterated local search', Kluwer Academic Publishers, pp.321–353, Norwell, MA.

Montgomery, D.C.(2004) 'Design and Analysis of Experiments', 6th edition, John Wiley & Sons, 660 p., New-York, USA.

Pepin, A.-S., Desaulniers, G., Hertz, A. and Huisman, D. (2008) 'Comparison of heuristic approaches for the multiple vehicle scheduling problem', *Journal of Scheduling (In press).*

Pullan, W. and Hoos, H.H. (2006) 'Dynamic local search for the maximum clique problem', *Journal of Artificial Intelligence Research*, Vol. 25, pp.159–185.

Oukil, A., Ben Amor, H., Desrosiers, J., and Gueddari, H. (2007) 'Stabilized column generation for highly degenerate multiple-depot vehicle scheduling problem', *Computers & Operations Research*, Vol. 3, No. 34, pp.817–834.

Ribeiro, C., and Soumis, F. (1994) 'A column generation approach to the multiple-depot vehicle scheduling problem', *Operations Research*, Vol. 42, No. 1, pp.41–53.

Rousseau, J.-M., Lessard, D., and Désilets, A. (1988) 'Aliages: a system for the assignment of bus routes to garages', In *Computer-Aided Scheduling of Public Transport*, pp.8–14, Berlin, Springer Verlag.

Tuza, Z. (1997) 'Graph colorings with local constraints - a survey', *Discussiones Mathematicae - Graph Theory*, Vol. 17, No. 2, pp.161–228.

Vasquez, M. (2002) 'Arc-consistency and tabu search for the frequency assignment problem with polarization', In: *Proceedings of CPAIOR'02*, pp.359-372, Le Croisic, France.