Available online at www.sciencedirect.com

SCIENCE DIRECT®

ELSEVIER

European Journal of Operational Research 174 (2006) 1117–1139

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/ejor

Discrete Optimization

A fast heuristic for solving a large-scale static dial-a-ride problem under complex constraints

Zhihai Xiang [a], Chengbin Chu [a,b,*], Haoxun Chen [a]

[a] *Laboratoire d'optimisation des systèmes industriels, Université de technologie de Troyes, 12 rue Marie Curie, BP 2060, 10010 Troyes Cedex, France*

[b] *School of Management, Hefei University of Technology, Tunxi Road No. 193, Hefei 230009, Anhui, China*

Received 26 February 2004; accepted 29 September 2004
Available online 23 June 2005

Abstract

This paper presents a heuristic, which concentrates on solving a large-scale static dial-a-ride problem bearing complex constraints. In this heuristic, a properly organized local search strategy and a diversification strategy are used to improve initial solutions. Then the improved solutions can be refined further by an intensification strategy. The performance of this heuristic was evaluated by intensive computational tests on some randomly generated instances. Small gaps to the lower bounds from the column generation method were obtained in very short time for instances with no more than 200 requests. Although the result is not sensitive to the initial solution, the computational time can be greatly reduced if some effort is spent to construct a good initial solution. With this good initial solution, larger instances up to 2000 requests were solved in less than 10 hours on a popular personal computer.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Transportation; Vehicle routing problem; Dial-a-ride problem; Heuristics

1. Introduction

In this paper, we consider a kind of dial-a-ride problem (DARP) which involves scheduling a heterogeneous vehicle fleet and a group of drivers with different qualifications based at a single depot to cover the transportation requests of customers. After negotiating with the agency, the customer specifies the pickup time window and the tolerable extra travelling time on the trip. Customers of different types may require different type of accommodations. Each type of vehicle varies in the number and the type of accommodations and it can only be driven by the corresponding qualified

[*] Corresponding author. Address: UTT-LOSI, 12 rue Marie Curie, BP 2060, 10010 Troyes Cedex, France. Tel.: +33 3 25 71 56 30; fax: +33 3 25 71 56 49.
E-mail address: chengbin.chu@utt.fr (C. Chu).

0377-2217/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2004.09.060

drivers. Drivers are under the constraints of the maximum driving duration of one trip, the break time between two successive trips and the upper bound of the accumulated driving time in one servicing horizon (e.g., one day). To the authors' knowledge, these real life constraints are seldom discussed together in literatures, especially the matching among customers, vehicles and drivers. The aim of the scheduling is to find the least cost trips to serve the maximum number of customers.

The DARP is a generalization of the capacitated pickup and delivery problem with time windows, which was first examined by Wilson et al. (1971). Thereafter, extensive studies were carried out over the past three decades (see the recent reviews of Desaulniers et al., 2002; Cordeau and Laporte, 2002). Since in practice the transportation requests are usually known in advance, most of the work has focused on static DARPs. However, a fast static algorithm is also very helpful for dynamic scenarios, where schedules are adjusted in real time.

Although the DARP is NP-hard (Healy and Moll, 1995), efforts were still made to get exact solutions. Early approaches (Psaraftis, 1980, 1983a,b) focused on solving single-vehicle problems using pure dynamic programming (DP) method. Then Desrosiers et al. (1986) introduced the concept of *dominance* to reduce intermediate states. This technique greatly speeds up the DP process if the problem subjects to strong constraints. Based on this work, some multi-vehicle problems can be exactly solved by the combination of the column generation (CG) method and the branch-and-bound process (Dumas et al., 1991). Instead of using the DP algorithm, some heuristics can also be used to get approximate solutions of the auxiliary subproblem (SP) in the CG framework (Savelsbergh and Sol, 1998; Xu et al., 2003). Thus, this method is capable of approximately solving large-scale and less strongly constrained problems.

Owing to the complexity of the DARP, the most popular approaches are still varieties of heuristics, which are usually characterized with two phases: a construction phase in which an initial schedule is obtained, and a tuning phase in which the solution is improved further.

In the construction phase, the techniques of sequential insertion (Jaw et al., 1986; Nanry and Barnes, 2000; Cordeau and Laporte, 2003) and parallel insertion (Roy et al., 1984a,b; Toth and Vigo, 1997) are commonly used, according to a certain criterion, e.g., the nearest distance or the minimum cost. The concept of *cluster first and route second* (Bodin and Sexton, 1986) may also be introduced into this step, in which the geographically close customers are grouped together before applying the single vehicle routing algorithm to each cluster. To overcome the difficulty arising from the dispersion of the two locations (pickup and delivery) represented by each customer, the use of mini-clusters is recommended (Dumas et al., 1989; Desrosiers et al., 1991; Ioachim et al., 1995).

In the tuning phase, the solution is improved by some local searches, which consist of intra-trip exchanges and inter-trip exchanges. The intra-trip exchanges adjust the sequence of stops within a single trip. The corresponding algorithms are usually based on Lin's (1965) λ-opt mechanism, which can be accomplished in $O(n^{\lambda})$ time (n is the number of stops in a trip). Several modifications were also developed, such as the algorithms of Lin and Kernighan (1973) and Or (1976). The inter-trip exchanges involve the exchange of stops among multiple trips. The details can be found in the work of Van Breedam (2001) and Kinderwater and Savelsbergh (1997).

Classical heuristics use only some local searches in a descent way, i.e., the value of the objective function at current iteration step is better than those at previous steps. The final result is usually observed as local optimum and is sensitive to the initial solution (Van Breedam, 2001). Modern heuristics, on the contrary, can temporarily accept some worse solutions during the optimization procedure. Thus, it is possible to drive the search out of local optima. The rules for accepting the worse solutions are termed *diversifications*. And these modern heuristics are often called *meta-heuristics*, for example, the simulated annealing (SA) (Kirckpatrick et al., 1983), the genetic algorithm (GA) (Holland, 1975), the Ant algorithm (AA) (Dorigo et al., 1991), the Tabu search (TS) (Glover and Laguna, 1997), the guided local search (GLS)

(Voudouris and Tsang, 1995), etc. The most popular meta-heuristic used for solving the DARP is probably the TS (Toth and Vigo, 1997; Nanry and Barnes, 2000; Cordeau and Laporte, 2003). By comparing different algorithms (Laporte et al., 2000; Van Breedam, 2001; Tan et al., 2001), it is observed that (1) descent algorithms usually generate local optima in a short time and are sensitive to initial solutions; (2) thanks to the diversification, meta-heuristics can generate 'global' optima and are insensitive to initial solutions; (3) meta-heuristics have no stopping criterion (usually stop after a number of iterations) and contain many case-sensitive empirical parameters, which may cause some difficulties for practical implementations.

Although many algorithms have been proposed for solving the DARP over the past three decades, new algorithms are still needed to meet the practical requirements. These new algorithms should be faster, simpler (with fewer empirical parameters) and more robust (Laporte et al., 2000). The goal of our current work is aimed at developing such a heuristic to generate acceptable solutions for large-scale real life applications.

It is noticed that the construction phase usually takes less than 1 second for the problem with hundreds of customers, while the tuning phase is the most time consuming part in the whole solving procedure. Therefore, the quality of the initial schedule and the efficiency of the tuning operation are the two critical aspects that should be concerned when developing a good algorithm. In this paper, a simple initial solution is proposed to give a good starting point, which speeds up the heuristic. And the tuning phase consists of a local search strategy, a diversification strategy and an intensification strategy. The local search strategy involves very simple but properly organized inter-trip exchanges, while the intra-trip exchange is accomplished simultaneously. Moreover, a concept of *margin time* is introduced to facilitate the local search and determines the starting time of a trip. The efficiency of the diversification strategy is guaranteed by a secondary objective function which points out a promising direction of diversification. Then by evaluating the *average wasted cost* of a single trip, the intensification strategy excludes some good trips from further refinement. Thus, the remaining trips get some chances to be refined further and many useless explorations are avoided.

The following sections of this paper are organized as: the problem is defined in Section 2; the heuristic is described in Section 3; computational results are detailed in Section 4; final conclusions are drawn in Section 5.

2. The problem

Practically, customer requests may be classified into outbound trips and inbound trips (Toth and Vigo, 1997). However, these two types of trips are treated as two indifferent requests in this heuristic. Therefore, we take into account only the requests of picking up one customer and delivering him or her to the destination. The depot, the pickup site and the delivery site are represented by a vertex set $\boldsymbol{V} = \{v_0, v_1, \ldots, v_{2n}\}$, where n is the number of requests; v_0 is the depot; v_i $(i = 1, 2, \ldots, n)$ denotes the pickup site of request i and v_{i+n} is the corresponding delivery site. In this paper, the traffic congestion is ignored. Hence, the travelling speed is constant. Three types of constraints are considered in the DARP interested here. They are:

Constraints on vertices:

C1: Every trip starts and ends at the depot v_0.

C2: The pair of v_i and v_{i+n} $(i = 1, 2, \ldots, n)$ belongs to the same trip (pairing constraint) and v_{i+n} is visited after v_i (priority constraint).

C3: At each vertex i $(i = 0, 1, \ldots, 2n)$, the arrival time is denoted as a_i and the service begins at time b_i, which is in the interval of the time window $[e_i, l_i]$; i.e., $b_i = \max\{a_i, e_i\}$ and $b_i \leqslant l_i$. At the depot v_0, e_0 and l_0 are the open time and close time of the agency, respectively. At the pickup site v_i $(i = 1, 2, \ldots, n)$, the time window is the result of the negotiation between the customer and the agency. If a_i is less than e_i, a waiting time is incurred; i.e., the waiting time is defined as max $\{e_i - a_i, 0\}$. At the delivery site v_{i+n}, we define $e_{i+n} \equiv e_i + s_i + t_{i,i+n}$, where s_i is the service time at v_i and $t_{i,i+n}$ is the direct travelling time from v_i to v_{i+n}. This definition implies that the unreasonable waiting time at the delivery site is not allowed.

Moreover, each customer specifies a tolerable extra travelling time t_i^{ext} beyond $t_{i,i+n}$. Thus, the time window at v_{i+n} is defined as $[e_{i+n}, e_{i+n} + t_i^{\text{ext}}]$. In addition, $t_i^{\text{ext}} \geqslant l_i - e_i$ is required to ensure that the time window at v_{i+n} will not be violated if $a_i = l_i$. Because the idle time inconvenience is incorporated in t_i^{ext}, the constraint of maximum waiting time at each vertex is not considered. Consequently, the waiting is not allowed in the time window. In this paper, all time windows are hard.

Constraints on vehicles:

C4: Each vehicle has limited number of accommodations for different types of customers (capacity constraint). Some accommodations may be suitable for more than one type of customers. Without loss of generality, customers are classified into different levels according to their types and vehicles have accommodations of the corresponding levels. Each customer can only take an accommodation on the same or upper levels.

Constraints on drivers:

C5: The duration of any trip should not exceed an upper bound.

C6: After each trip the driver must take a break before executing the next task.

C7: The working time (including driving time, waiting time and break time) per day should not exceed an upper bound.

C8: According to their qualifications, drivers are also classified into the same levels as those of the customers. A driver can drive only a vehicle, which has the accommodations of the same or lower levels.

The cost of a trip is calculated as

$$C = C_{\text{fix}} + C_{\text{mileage}} + C_{\text{drive}} + C_{\text{wait}} + C_{\text{service}}, \tag{1}$$

where C_{fix} is the fixed cost of this trip; C_{mileage} is the mileage rate × total mileage of this trip; C_{drive} = driving rate × total driving time of this trip; C_{wait} = waiting rate × total waiting time of this trip; C_{service} = service rate × total service time of this trip.

C_{fix} and the mileage rate varies with different type of vehicles. Different drivers have different driving rates, waiting rates and service rates. The object function of this problem is the sum of the costs over all trips.

3. The heuristic

Before plunging into the details of this heuristic, how to determine the starting time of a trip should be discussed. It is clear that the earlier to start, the better to avoid violating the time window constraint. A simple way is to set the departure time from the depot as $e_{1\text{st}} - t_{0,1\text{st}}$ (this time should not be less than e_0), where 1st is the number of the first pickup site in the trip. However, it is possible to shrink the waiting time and the total duration of the trip by delaying the starting time. To achieve this, the techniques proposed by Savelsbergh (1992) and Hunsaker and Savelsbergh (2002) can really help. However, because the maximum waiting time at each vertex is not considered here, a similar but simpler method is introduced in the current work. This method involves two steps. In the first step, set $a_{1\text{st}} = e_{1\text{st}}$, then sequentially calculate and keep the arrival time at each vertex up to the last one in the trip. The task of the second step is to calculate and keep the margin time (MT) from the last vertex of the trip back to the first one. The MT at a vertex is defined as the maximum delay of the arrival time at this vertex, and this delay will not cause the violation of the time windows at the following vertices. Let m_i denote the MT at vertex i, it is calculated as:

$$m_i = \min\{b_i + m_{\text{next}}, l_i\} - a_i, \tag{2}$$

where m_{next} is the MT of the next vertex in the trip. In the beginning, $m_{\text{next}} = +\infty$. Thus, the starting time of the trip is determined as $e_{1\text{st}} - t_{0,1\text{st}} + m_{1\text{st}}$.

This heuristic is composed of a pre-processing phase to prepare some frequently used constants in advance, a construction phase to obtain an initial solution, an improving phase to improve the solution and an intensification phase to fine-tune the solution. The framework of this heuristic is depicted in Fig. 1, which is explained in the following.

3.1. Pre-processing

To avoid the redundant computation, some frequently used data are pre-calculated and stored in advance.

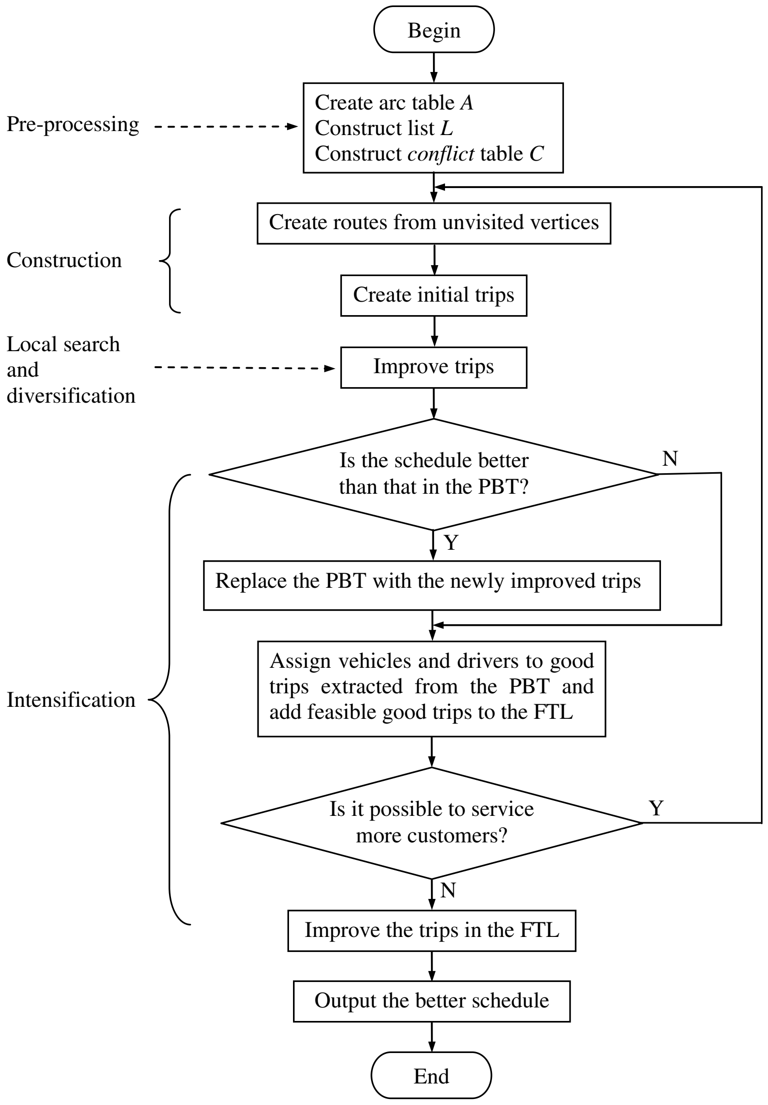
Begin
Pre-processing
Create arc table A
Construct list L
Construct conflict table C
Construction
Create routes from unvisited vertices
Create initial trips
Local search and diversification
Improve trips
Is the schedule better than that in the PBT?
N
Y
Replace the PBT with the newly improved trips
Intensification
Assign vehicles and drivers to good trips extracted from the PBT and add feasible good trips to the FTL
Is it possible to service more customers?
Y
N
Improve the trips in the FTL
Output the better schedule
End

Fig. 1. The framework of the heuristic.

First, an arc table $\boldsymbol{A}$ is created to keep the distance and the travelling time of admissible arcs. The inadmissible arcs are detected by examining the constraints of the pairing, the priority, the vehicle capacity, the time windows, the time windows and pairing of requests (see Dumas et al., 1991), and the maximum driving duration. This treatment, which takes $O(n^2)$ time, greatly improves the efficiency of the later operations.

Then a list $\boldsymbol{L}$ and a *conflict* table $\boldsymbol{C}$ are constructed for creating initial rough schedules. Details are described in the following subsection.

3.2. The construction phase

The main objective in this phase is constructing a feasible initial schedule without spending too much effort. Sure, the initial schedule of one trip servicing only one customer (OTOC) is the simplest. However, this initial schedule is usually so bad that too much effort will be spent to improve it. Therefore, additional effort is worthwhile for obtaining a better one. To construct such an initial solution, two problems should be solved: how to determine the sequence of vertices in a trip and how to group customers into different trips.

In this heuristic, the sequence of vertices in an initial trip is determined by the late time of the time windows imposed on them. The list ***L*** is constructed for this purpose. It contains the numbers of all vertices except for the depot. These numbers are sorted in ascending order with respect to their late times. This list can be constructed in no more than $O(n^2)$ time.

The customers are grouped into different trips in two steps according to the constraints (see Fig. 2).

Step 1: The time window constraint is used to cluster the customers into different groups. Let suffix '+' represent the pickup site and suffix '−' represent the delivery site. In a partial trip, there are three possible sequences, which contains customer i and j, i.e., $i+ \to j+ \to i- \to j-$, $i+ \to j+ \to j- \to i-$ and $i+ \to i- \to j+ \to j-$ (assume $l_{i+} \leqslant l_{j+}$). Because the sequence of these vertices is determined by their late times, actually there is only one possibility, e.g. $i+ \to j+ \to i- \to j-$. If the time window of $j+$ is not violated with $a_{i+} = l_{i+}$; the time window of $i-$ is not violated with $a_{j+} = l_{j+}$ and the time window of $j-$ is not violated with $a_{i-} = l_{i-}$, then it is said that customer i and j can be serviced in one trip without violating the time window constraint in the worst case. Otherwise customer i and customer j are marked *conflict*. This information is kept in the *conflict* table ***C***. According to the list ***L*** and the *conflict* table ***C***, unvisited vertices are clustered into different groups, called *routes*, and each pair of customers in the same route does not *conflict* with each other. This operation, which takes $O(n^2)$ time, is described in Fig. 3.

Step 2: Other constraints are used to create initial trips from each route. One elementary operation involved in this step is constructing a feasible trip by sequentially extracting as many as possible customers from one route. The feasibility examination covers all the rest constraints. Instead of actually assigning the vehicle and the driver, only the types of the available vehicle and the driver are considered here. After extracting one trip, the remaining vertices in the route are regrouped into a new route. Trips are continuously generated until there is no customer left. This procedure is similar to the *sweep* algorithm of Gillett and Miller (1974).

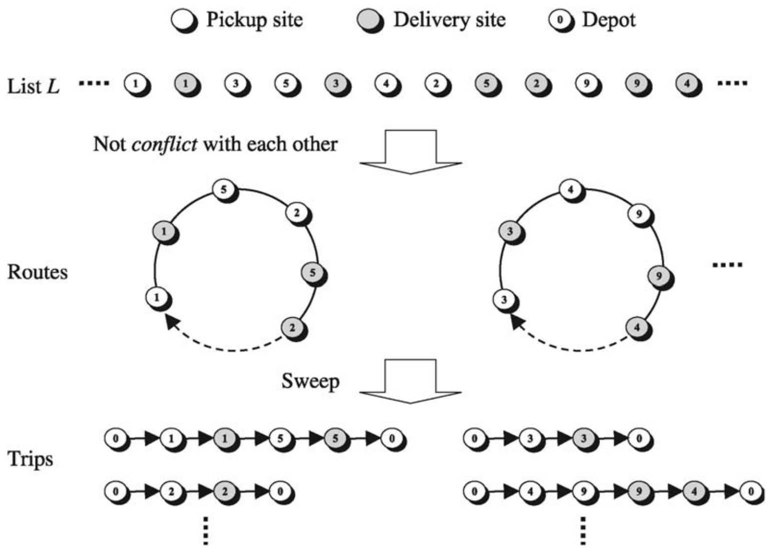

Fig. 2. The construction of initial trips.

```
Procedure Create_route
For each unvisited vertex $i$ in list $L$ Do
    If vertex $i$ is a pickup site Then
        Add vertex $i$ as the first pickup site in a new route.
        For each unvisited vertex $j$ after vertex $i$ in list $L$ Do
            If vertex $j$ is a pickup site and does not conflict with any customer
            already in this route or vertex $j$ is a delivery site and its
            corresponding pickup site is already in this route Then
                Add vertex $j$ to the tail of this route.
            End If
        End For
    End If
End For
End
```

Fig. 3. The algorithm of creating routes.

The reason why the *conflict* table $\boldsymbol{C}$ is constructed in the worst case can be explained as follows. Supposing the partial sequences of $i+ \to j+ \to i- \to j-$, $i+ \to i- \to k+ \to k-$ and $j+ \to k+ \to j- \to k-$ are all feasible with $a_{i+} = e_{i+}$ and $a_{j+} = e_{j+}$. Thus, customer i, j and k are grouped in a route, say $i+ \to j+ \to i- \to k+ \to j- \to k-$. However, the time window of $j-$ is likely to be violated because $k+$ is inserted between $i-$ and $j-$ or $i-$ is inserted between the $j+$ and $k+$. Therefore, the time window constraint should also be considered in Step 2 to construct feasible trips. This examination is very complex and invalidates Step 1. However, if the *conflict* is examined in the worst case, the time window constraint is surely observed in each route and constraint examination in Step 2 can be greatly simplified.

This initial solution can be conveniently called as a cluster first sweep second (CFSS) initial solution.

3.3. The improving phase

The improving phase consists of a properly organized local search strategy to explore the local region and a diversification strategy to drive the search out of local minima.

3.3.1. The local search strategy

Three simple elementary local searches are properly organized (see Fig. 4) to improve the solution in local regions:

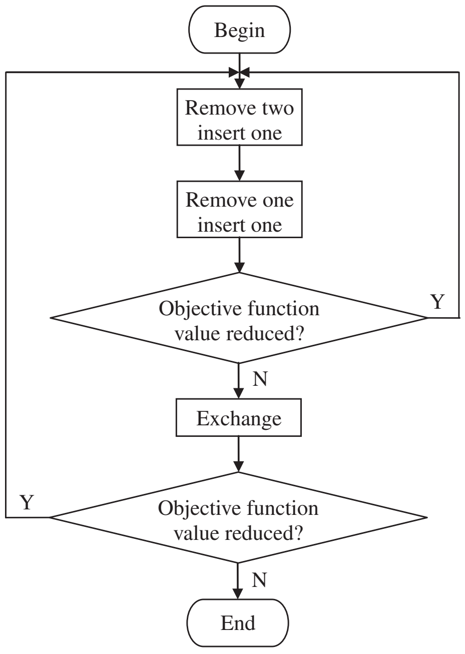
Begin
Remove two insert one
Remove one insert one
Objective function value reduced?
Y
N
Exchange
Objective function value reduced?
Y
N
End

Fig. 4. The local search strategy.

(1) The local search of ''remove two insert one'' (see Fig. 5(a)) tries to remove two customers from one trip and inserts them to another trip one by one. The number of trips can possibly be reduced by this operation. It takes about O(n^2) time (n is the number of customers in this problem).
(2) The local search of ''remove one insert one'' (see Fig. 5(b)) tries to remove one customer from one trip and insert it to another trip. This operation is also possible to reduce the number of trips. It is approximately in O(n) time.
(3) The local search of ''exchange'' (see Fig. 5(c)) tries to swap two customers between two trips. This operation cannot reduce the number of trips. It is approximately in O(n^2) time.

All elementary local searches involve some *moves* between two trips (e.g., trip 1 and trip 2 in Fig. 5). If this *move* successfully reduces the value of the objective function, the *move* among trip 1 and other trips will not be considered, i.e., the *first improvement* is adopted here. According to the study of Van Breedam (2001), there is no significant difference between the *first improvement* and the *best improvement*.

Two frequently used elementary operations in the local search are ''removing a customer from a trip'' and ''inserting a customer into another trip''. The removal operation is simple and straight forward, while the insertion operation is more complicated. In the current work, the *best insertion* is adopted, i.e., the pickup site and delivery site of a customer are inserted to the optimal positions in

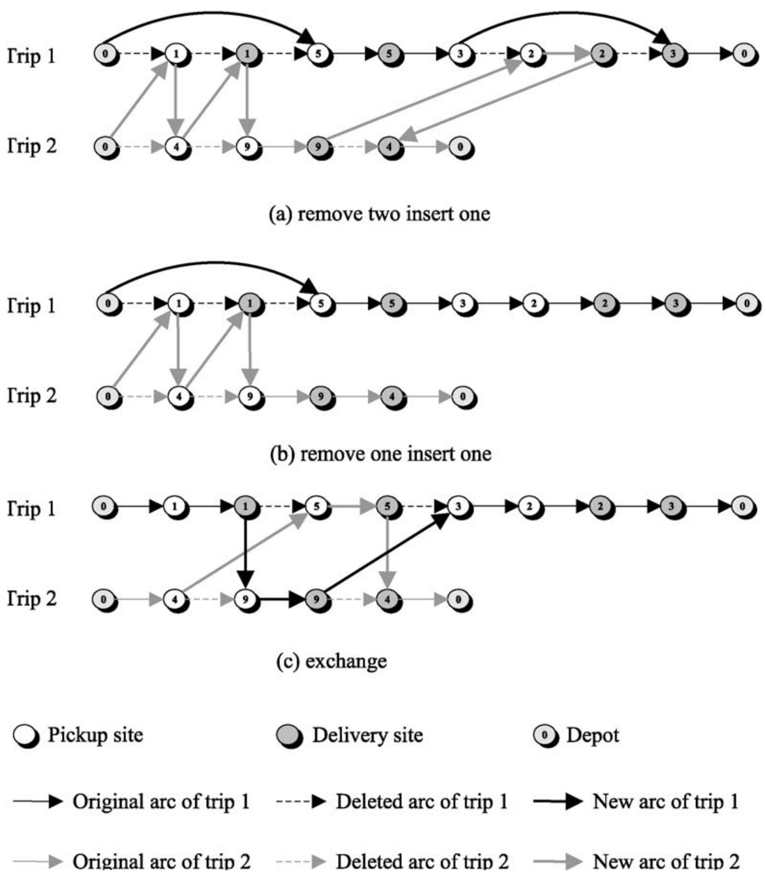

Fig. 5. Elementary local searches.

a trip yielding the least value of the objective function. If there are r vertices in the trip, this operation takes $O(r^2)$ time. However, according to the arc table $\boldsymbol{C}$ and the pre-calculated MT at each vertex, it is very easy and fast to detect whether a vertex can be inserted to that position. Moreover, the maximum driving time constraints should not be violated after the insertion.

Moreover, when it fails to insert one customer into another trip, the policy of *best insertion* will also be adopted to insert this customer back to its original trip. Therefore, the sequence of vertices in the original trip could be changed. This acts just like an intra-trip exchange. Thus, in this heuristic the inter-trip exchange and the intra-trip exchange are carried out in parallel, which greatly improves the efficiency of the local search.

Once the vertices are changed in a trip, all information of this trip is updated. Here again, only the types of the available vehicle and the driver are interested.

3.3.2. The diversification strategy

The diversification is introduced to guide the local search to other regions, where better solutions probably exist. The direction of the diversification can be determined at random according to some specified probabilities, such as the acceptance rate in the SA, the mutation rate in the GA and the exploitation rate in the AA. Other metaheuristics, such as the TS and the GLS (Backer et al., 2000), prohibit the local search from entering unpromising regions by using some penalties (the Tabu list and the augmented cost function). In this heuristic a secondary objective function (SOF) is used to guide the local search to some promising regions, where better solutions are likely to be found and no empirical parameter need be specified.

The idea of using SOF can be traced back to the work of Healy and Moll (1995). They called the SOF as the *sacrificing metric*, which was constructed according to the cost and the neighbourhood size of the local minima and intended to give large neighbourhoods. Based on the same idea, Cordone and Wolfler Calvo (2001) constructed a heuristic for the vehicle routing problem with time windows. They selected the total trip duration as the SOF based on the criterion that the SOF "has to be rather different from the main one in order to drive the search far enough... must be partially dependent on the first one, in order to avoid worsening its value too strongly". Both of these papers explain that by using the SOF the primary objective function (POF) deteriorates temporarily but better solutions can be found later because the local search neighbourhood changes. That is to say using the SOF improves the POF indirectly.

However, we will show in this paper that using a proper SOF can also improve the POF directly. In our opinion, an ideal SOF should light a different path that directs to the same global minimum as expected by the POF. However, it could be very difficult or impossible to find this ideal SOF for a specific problem and usually a good one has to be used instead.

It is conceived that a good trip should fully exploit the resources. Unfortunately, in practice every trip has some *idle states*, where there is no customer on board. Since customers cannot directly benefit from *idle states*, the cost incurred in these periods is regarded to be wasteful. The wasted cost of a trip has to be shared by all the customers involved. Therefore, the average wasted cost (AWC) is an important property of a trip. This property is calculated as:

$$\mathrm{AWC} = \frac{C_{\mathrm{fix}} + C^{\mathrm{idle}}_{\mathrm{mileage}} + C^{\mathrm{idle}}_{\mathrm{drive}} + C^{\mathrm{idle}}_{\mathrm{wait}}}{\text{the number of customers}}, \tag{3}$$

where $C^{\mathrm{idle}}_{\mathrm{mileage}}$ = mileage rate $\times$ total idle mileage of this trip; $C^{\mathrm{idle}}_{\mathrm{drive}}$ = driving rate $\times$ total idle driving time of this trip; $C^{\mathrm{idle}}_{\mathrm{wait}}$ = waiting rate$\times$ total idle waiting time of this trip.

Actually, the fixed cost is not a wasted cost. The reason of including it in the AWC is that trips with lower fixed cost are more preferable to those with higher fixed cost, because the costlier vehicles should be used more efficiently than the cheaper ones.

The AWC can be used to evaluate the quality of a trip only if this trip has been optimized by some intra-trip local searches. For example as Fig. 6 shows, if time windows are wide enough, both trip 1 and trip 2 are possible solutions. The AWC of

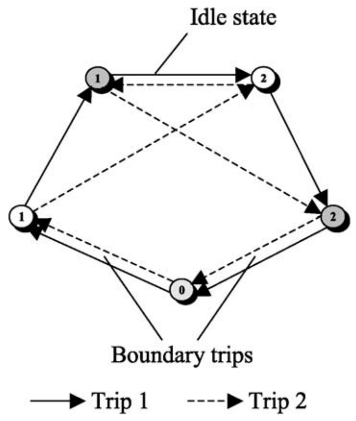

Fig. 6. An example of a worst trip with a better AWC.

trip 2 is obviously less than that of trip 1. However, the trip 1 has shorter mileage and is more likely to have a lower cost. Fortunately, only trip 1 will 'survive' after applying the *best insertion* during the local search.

Although the AWC is a good index of the quality of a trip, the AWC over all trips is not a good choice for the SOF. Because it has something in common with the POF and these common parts have weak effect on diversification. One common part is the C_{fix} and another one is the idle cost occurred in the boundary trips, which are the trip from the depot to the first pickup site and the trip from the last delivery site back to the depot (see Fig. 6). Excluding these common parts and adding a penalty for detours, a modified wasted cost (MWC) is defined as:

$$\text{MWC} = \alpha \times \left(MC^{\text{idle}}_{\text{mileage}} + MC^{\text{idle}}_{\text{drive}} + C^{\text{idle}}_{\text{wait}} \right), \tag{4}$$

where $MC^{\text{idle}}_{\text{mileage}} = \text{mileage rate} \times (\text{total idle mileage of this trip} - \text{the mileage of boundary trips})$; $MC^{\text{idle}}_{\text{drive}} = \text{driving rate} \times (\text{total idle driving time of this trip} - \text{the driving time of boundary trips})$; $\alpha = \frac{\sum \text{actual travelling time of each customer}}{\sum \text{expected travelling time of each customer}}$.

The travelling time of a customer is the time period starting from the beginning of the pickup service to the end of the delivery service. And 'expected' means that this customer travels from its pickup site directly to its delivery site. If there is a detour, α is greater than one and serves as a penalty.

The MWC over all trips is used as the SOF in this heuristic. Incorporating this SOF, two local searches are iteratively conducted to improve the trips (see Fig. 7). The iteration stops when no trip was deleted and

$$\frac{\text{min\_cost} - \text{total\_cost}}{\text{min\_cost}} < 5‰, \tag{5}$$

where min_cost is the minimum value of the POF ever found and total_cost is the freshly obtained value of the POF. When there are some trips deleted, there could be some chances to reduce the total_cost more than 5‰ in the next iteration. Therefore, temporarily accepting the solution due to deleting trips can also be regarded as a diversification mechanism.

The following three examples illustrate in detail why this diversification strategy is helpful.

Example 1. Supposing state 1 contains

trip 1: $0 \to 1+ \to 1- \to 2+ \to 2- \to 0$,
trip 2: $0 \to 3+ \to 3- \to 0$ and,
trip 3: $0 \to 4+ \to 4- \to 5+ \to 5- \to 0$.

State 2 contains

trip 1′: $0 \to 5+ \to 5- \to 0$,
trip 2′: $0 \to 3+ \to 3- \to 1+ \to 1- \to 0$ and,
trip 3′: $0 \to 4+ \to 4- \to 2+ \to 2- \to 0$.

State 2 is a better solution. If any *move* between two trips of state 1 increases the POF, it is impossible to reach state 2 from state 1 with the local search strategy in this heuristic. However, if the SOF can be successively reduced by swapping customer 2 and customer 3 then swapping customer 2 and customer 5, state 2 can be reached by conducting this local search strategy. In this example, the schedule can be improved directly by using the SOF.

Example 2. Supposing state 1 contains

trip 1: $0 \to 1+ \to 1- \to 0$,
trip 2: $0 \to 2+ \to 2- \to 0$ and,
trip 3: $0 \to 3+ \to 3- \to 4+ \to 5+ \to 4- \to 5- \to 0$.

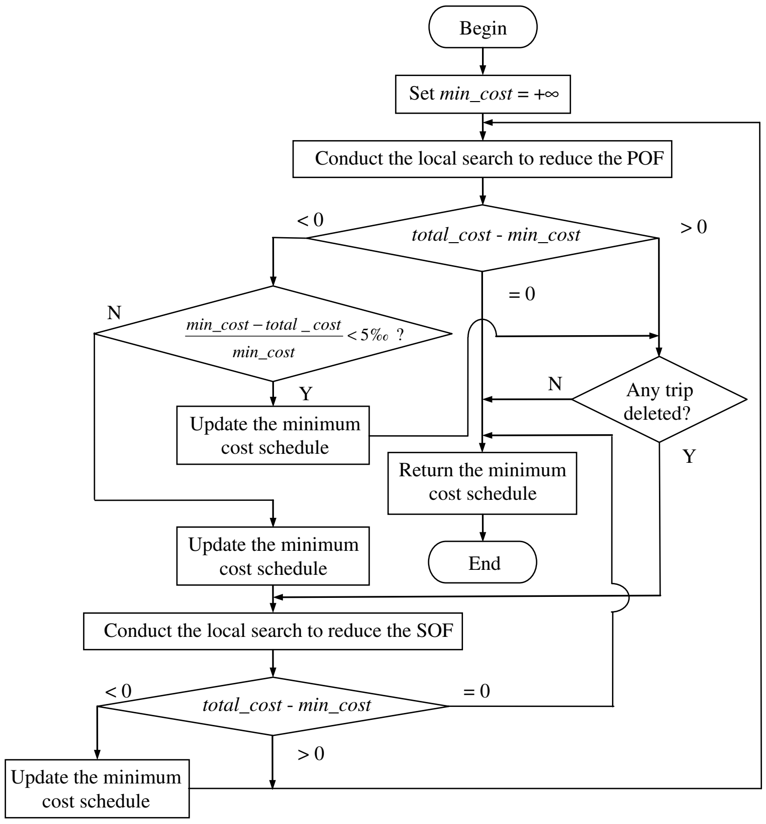
Begin
Set min_cost = +∞
Conduct the local search to reduce the POF
total_cost - min_cost
< 0
> 0
= 0
N
min_cost − total_cost / min_cost < 5‰ ?
Y
Update the minimum cost schedule
N
Any trip deleted?
Y
Return the minimum cost schedule
End
Update the minimum cost schedule
Conduct the local search to reduce the SOF
< 0
total_cost - min_cost
= 0
> 0
Update the minimum cost schedule

Fig. 7. The strategy of improving trips.

State 2 contains

trip 1′: $0 \to 4+ \to 4- \to 1+ \to 1- \to 0$,

trip 2′: $0 \to 2+ \to 2- \to 5+ \to 5- \to 0$ and

trip 3′: $0 \to 3+ \to 3- \to 0$.

State 2 is a better solution and state 1 is obtained after conducting the local search with the POF. If α is not included in Eq. (4), the *move* of “removing customer 4 from trip 3 and inserting it into trip 1” and the *move* of “removing customer 5 from trip 3 and inserting it into trip 2” may probably increase the SOF. Therefore, it may be impossible to reach state 2 by conducting the local search with the SOF. However, with the help of α, the WMC of trip 3 could be very large and the state 2 could be readily obtained.

Example 3. Supposing state 1 contains

trip 1: $0 \to 1+ \to 2+ \to 1- \to 2- \to 0$,
trip 2: $0 \to 3+ \to 3- \to 0$ and,
trip 3: $0 \to 4+ \to 4- \to 0$.

State 2 contains

trip 2′: $0 \to 3+ \to 3- \to 1+ \to 1- \to 0$ and,
trip 3′: $0 \to 4+ \to 4- \to 2+ \to 2- \to 0$.

State 2 is a better solution. It is impossible to reach state 2 from state 1 by conducting the local search with the POF. According to the diversification strategy in this heuristic, the local search with the SOF is conducted next. During this local search, if neither customer 1 nor customer 2 can be moved to other trips from trip 1, they will be inserted back to trip 1 at the best positions. Since there is no *idle state*, according to Eq. (4) the MWC of trip 1 is zero. If the MWC is zero, the first possible position is regarded as the best position. Therefore, trip 1′: $0 \to 1+ \to 2+ \to 2- \to 1- \to 0$ is obtained. Because trip 1 is the result from the previous local search with the POF, where the *best insertion* is also applied, trip 1′ must be more expensive than trip 1. Thereafter in the next local search with the POF, moving customer 1 from trip 1′ to trip 2 then moving customer 2 to trip 3 will possibly be successful *moves*. Thus, state 2 is reached indirectly.

Because this heuristic utilizes only very simple elementary local searches, it cannot explore larger neighbourhoods, which can be covered only by expensive exchanges involving more than two trips at the same time. However, as the above three examples (other more complex examples are not enumerated here) illustrate, the 'terrain' of the local search region can be properly changed by using a proper SOF so that many "dead corners" can also be efficiently swept directly or indirectly.

3.4. The intensification phase

After the above improvement, a good schedule is obtained and the improved trips are called probable best trips (PBT). Giving them the attribute of "probable best" is due to two reasons. First, because only the types of the available vehicle and the driver are assigned to each trip, if there are more trips than the actual vehicles and drivers some trips are infeasible. Second, this good schedule is still possible to be refined further.

As stated in Section 3.3.2 the AWC can be used to evaluate the quality of a trip. If some trips were already of good quality, there could be little potential to improve them further. These good trips should be extracted from the PBT and have high priority to get their vehicles and drivers. Then more effort should be spent to fine-tune the remaining trips.

A trip is regarded to be good if its AWC is less than a threshold. The threshold is specified according to a global average wasted cost (GAWC), which is the AWC over all customers in the PBT. It can simply take the GAWC as the threshold, which is called the *large threshold*. Alternatively, a *small threshold* can be specified as:

$$\text{small threshold} = \frac{\text{GAWC}}{\beta}, \quad \beta \geqslant 1, \tag{6}$$

where β is initially set to be 2.0, then it is dynamically changed. How to change β will be explained later.

The rule of maximum spare time block (MSTB) is applied when assigning a vehicle and driver to a trip. As Fig. 8 shows, vehicle 1 and vehicle 2 are feasible candidates for servicing trip k. After inserting trip k, the MSTB of vehicle 1 is less than that of vehicle 2. Therefore, vehicle 2 is assigned to trip k. Thus, vehicle 1 is more capable of servicing other trips. The same rule is applied for assigning drivers, but the constraints of the break time and the maximum working time should be considered for selecting feasible candidates. It is possible that some good trips are infeasible due to the lack of vehicles or drivers. However, this does not definitely imply that the customers in these trips cannot be serviced. If these customers are rescheduled together with the customers in the remaining 'bad' trips of the PBT, feasible services may be provided for them. Therefore, these infeasible good trips will be inserted back to the PBT.

All feasible good trips are added into a feasible trip list (FTL), then the remaining customers in the PBT are rescheduled from the beginning; i.e., a DARP with fewer customers and fewer resources of vehicles and drivers will be solved with the algorithms presented in previous sections. During the new optimization procedure, it is possible to find a better schedule for these customers, because a new initial solution is constructed and the neighbourhood changes. If it is so, the PBT is replaced with these newly improved trips and β is reset to be 2.0. If it fails to find a better schedule, these newly improved trips are discarded and if the previous value of β is 2.0, β is set to be 1.5; otherwise β is set to be 1.0. Then the extraction of good trips and the rescheduling of the remaining customers in

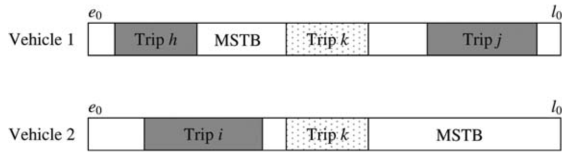

Fig. 8. The illustration of the MSTB.

the PBT are iteratively carried out until all customers have been serviced or no customer can be serviced due to the lack of vehicles or drivers (see Fig. 1).

The last chance of intensification is taking the trips in the final FTL as an initial solution and trying to improve them by using the strategy described in Fig. 7. After assigning another copy of vehicles and drivers to all improved trips, a new schedule is obtained. If this new schedule services more customers at less cost than the old schedule, it will be taken as the final solution. Otherwise, the old schedule wins out.

4. Computational experiments

Although DARP has been extensively studied, to the authors' knowledge, there is no benchmark for the problem with the same constraints considered in this paper. Therefore, our only choice to evaluate this heuristic was comparing the results with the lower bound obtained from the CG method. The CG algorithm adopted here almost follows the work of Dumas et al. (1991). However, small modifications were made, because the problems are different in the objective function and constraints. The modified CG algorithm is presented in Appendix A.

The heuristic and the CG algorithm have been coded into two standard C programs, which are optimized for the maximum speed by the Microsoft C compiler. All the following computations were carried out on a DELL GX260 PC with 2.4 G Pentium IV processor and 1 GB memory under the Windows 2000 environment. All codes ran in the memory without swapping.

4.1. Configuration of instances

The agency provides the service over the region of 40×50 km from 6:00 to 18:00. There are three types of customers, vehicles and drivers. The parameters of different type of them are listed in Table 1.

For every instance, the positions of the depot, pickup sites and delivery sites are uniformly generated at random over the service region. The type of each customer is also randomly generated with the probabilities of 50% for type I, 25% for type II and 25% for type III, respectively. The time windows are specified as follows (the vertex number of the pickup site is denoted as i and the vertex number of the delivery site is denoted as $i + n$):

Table 1
The parameters of customers, vehicles and drivers

		Type I	Type II	Type III
Customer	Service time (minutes)	3	4	5
	Maximum time window at pickup site (minutes)	15 (30)	15 (30)	15 (30)
	Minimum tolerable extra travelling time (minutes)	20 (30)	20 (30)	20 (30)
	Maximum tolerable extra travelling time (minutes)	30 (60)	30 (60)	30 (60)
Vehicle	Speed (kilometer/hour)	36	36	36
	Fixed cost	100	125	150
	The number of accommodation type I	4	2	2
	The number of accommodation type II	0	2	0
	The number of accommodation type III	0	0	2
	Mileage rate (per km)	2	3	3
Driver	Maximum trip duration (minutes)	510	510	510
	Break time (minutes)	105	105	105
	Maximum working time (minutes)	600	600	600
	Driving rate (per hour)	42	48	54
	Waiting rate (per hour)	39	45	51
	Servicing rate (per hour)	39	45	51

Note: The parameters in the parentheses are of the wide time window instances.

$$e_i = e_0 + t_{0,i} + \text{rand} \quad \times (l_0 - e_0 - t_{0,i}), \tag{7}$$

$$l_i = e_i + s_i + \text{rand} \times (\text{max}_\text{w} - s_i), \tag{8}$$

$$t_i^{\text{ext}} = \text{min\_ext} + \text{rand} \times (\text{max\_ext} - \text{min\_ext}), \tag{9}$$

where max_tw represents the maximum time window at the pickup site; min_ext represents the minimum tolerable extra travelling time; max_ext represents the maximum tolerable extra travelling time; and rand is a uniform pseudo-random number in the interval of $[0, 1]$, which simulates the negotiation between the customer and the agency. Euclidean distances are adopted to calculate the distance between two vertices. All the values generated, if fractional, are rounded into the closest integers. To ensure the service of each single customer is feasible, if $e_{i+n} + s_i + t_{i+n,0} > l_0$ or $e_{i+n} + s_i + t_{i+n,0} - e_i + t_{0,i} >$ maximum driving time, all information of this request will be regenerated.

Two groups of instances with the time window of different width (see Table 1) were generated. Each group contains seven classes of instances with the number of customers ranging from 50 to 2000. Ten instances were generated for each class.

4.2. Computational results

Both the CFSS and the OTOC initial solutions were used to examine the impact of the initial solution. The influences of the *large threshold* and the *small threshold* were also evaluated. Therefore, each instance was solved by four programs, which are different in the initial solution and the threshold.

Firstly, the instances with no more than 200 customers were calculated. The computational times are depicted in Figs. 9 and 10. It reveals that the program with the OTOC is always slower than the program with the CFSS and the program with the *small threshold* is always more time consuming than that with the *large threshold*. However, all instances were solved very quickly even with the slowest program.

The solutions are compared with the lower bounds obtained by the modified CG method.

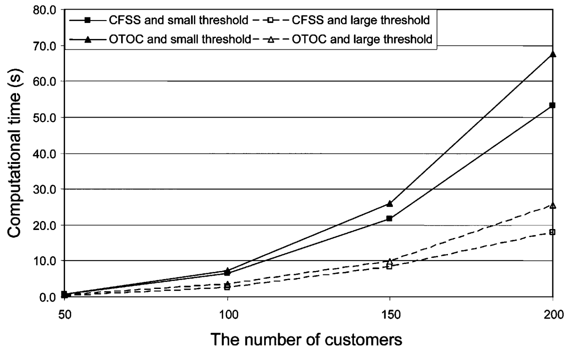

Fig. 9. The computational time of narrow time window instances.

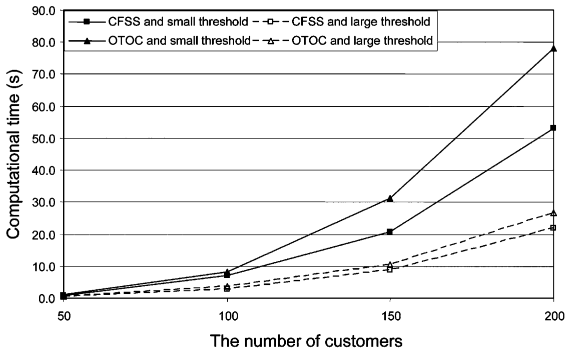

Fig. 10. The computational time of wide time window instances.

The average gaps to the lower bounds of different solutions are compared in Figs. 11 and 12. Detail

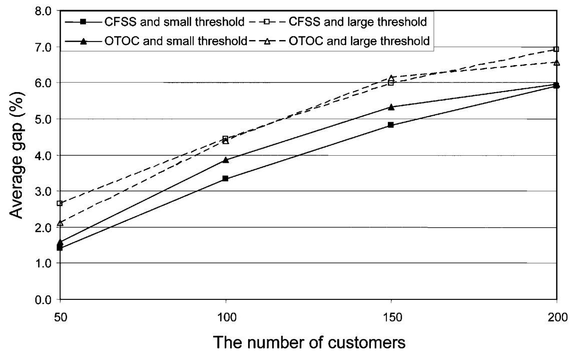

Fig. 11. Computational results of narrow time window instances.

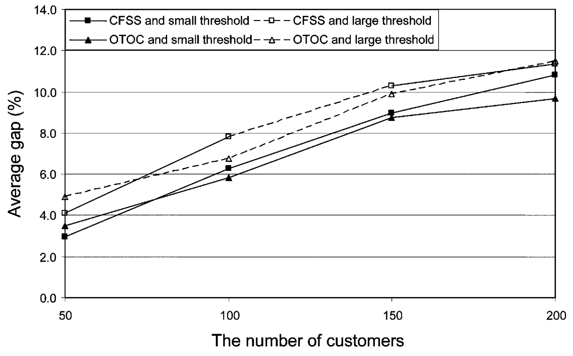

Fig. 12. Computational results of wide time window instances.

comparisons are listed in Tables 2 and 3. It is noticed that the average gaps increase when the number of customers increases. However, because the tightness of the gap from CG method deteriorates with the increase of customers, these results could convince us that these solutions have acceptable quality. Furthermore, it is observed that there is not so much difference between the results from different initial solutions and better results are obtained with the *small threshold*. These two aspects will be discussed later in detail.

To evaluate the benefits from the diversification and the intensification, a diversification gain (DG) and an intensification gain (IG) are evaluated. They are defined as:

$$DG = \frac{C_{\text{div}} - C_{\text{local}}}{C_{\text{local}}} \times 100\%, \tag{10}$$

$$IG = \frac{C_{\text{final}} - C_{\text{local}}}{C_{\text{local}}} \times 100\% - DG, \tag{11}$$

where C_{local} is the value of the POF by conducting the pure local search; C_{div} is the value of the POF obtained by the improving operations (see Fig. 7); and C_{final} is the ultimate value of the POF. Comparing with the gaps to the lower bounds (see Tables 2 and 3), it is very obvious that the diversification and the intensification play very important roles in this heuristic.

Table 2
Computational results of narrow time window instances

Initial solution	Threshold		Total number of customers			
			50	100	150	200
CFSS	Small	Average gap (%)	1.4[a]	3.3[a]	4.8[a]	5.9[a]
		Maximum gap (%)	2.1	5.3	6.6	6.8
		Average IG (%)	1.8	1.2	1.8	1.3
		Average CPU time (seconds)	0.7	6.5	21.8	53.3
	Large	Average gap (%)	2.7	4.5	6.0	6.9
		Maximum gap (%)	4.3	6.6	7.5	8.3
		Average IG (%)	0.6	0.2	0.7	0.4
		Average CPU time (seconds)	0.2	2.5	8.3	18.0
	Small and large	Average DG (%)	0.9	2.9	1.4	2.8
OTOC	Small	Average gap (%)	1.6	3.9	5.3	6.0
		Maximum gap (%)	3.2	5.0	7.8	7.3
		Average IG (%)	0.8	1.1	1.3	0.9
		Average CPU time (seconds)	0.7	7.2	25.9	67.7
	Large	Average Gap (%)	2.1	4.4	6.1	6.6
		Maximum gap (%)	3.7	6.6	9.5	7.4
		Average IG (%)	0.3	0.6	0.6	0.4
		Average CPU time (seconds)	0.4	3.5	9.9	25.6
	Small and large	Average DG (%)	1.1	2.6	2.2	3.1

[a] The smallest gap in this column.

Table 3
Computational results of wide time window instances

Initial solution	Threshold		Total number of customers			
			50	100	150	200
CFSS	Small	Average gap (%)	2.9[a]	6.3	9.0	10.8
		Maximum gap (%)	4.6	10.0	10.7	14.0
		Average IG (%)	1.8	1.9	2.7	1.9
		Average CPU time (seconds)	0.8	7.1	20.7	53.2
	Large	Average gap (%)	4.1	7.8	10.3	11.4
		Maximum gap (%)	12.2	10.1	14.2	14.0
		Average IG (%)	0.8	0.5	1.5	1.4
		Average CPU time (seconds)	0.5	2.7	8.7	22.0
	Small and large	Average DG (%)	2.0	2.1	1.5	3.0
OTOC	Small	Average gap (%)	3.5	*5.8	*8.7	*9.6
		Maximum gap (%)	6.5	7.9	10.6	11.7
		Average IG (%)	2.3	2.2	1.8	2.0
		Average CPU time (seconds)	1.0	8.3	31.1	78.2
	Large	Average gap (%)	4.9	6.8	9.9	11.5
		Maximum gap (%)	10.6	9.3	12.1	14.4
		Average IG (%)	1.1	1.4	0.8	0.4
		Average CPU time (seconds)	0.6	3.7	10.6	26.8
	Small and large	Average DG (%)	2.5	2.6	1.7	3.6

[a] The smallest gap in this column.

Encouraged by the above results, the heuristic was tested further by the instances with large number of customers. The computational results (because the CG method is very time consuming, the lower bounds for these instances cannot be obtained) are listed in Table 4, which shows that these large instances can be solved in a few hours. For example, with the CFSS initial solution and the large threshold, the instances with 2000 customers can be solved within 10 hours, which meets the requirement of daily scheduling.

Table 5 compares the results from different initial solutions. It is observed that the schedule costs are almost the same whether using the CFSS initial solution or the OTOC initial solution. The maximum difference is only 1.2%. This means the diversification is so powerful that this heuristic is not sensitive to the initial solution. However, the computational time can usually be greatly saved if using the CFSS initial solution. This is probably because the quickly obtained (only about 100 millisecond for instances with 2000 customers) CFSS initial solution is closer to the optimal solution than the OTOC initial solution.

As expected, the heuristic with the *large threshold* runs much faster than that with the *small threshold* (see Table 6, Figs. 9 and 10). Although the *large threshold* generates a gap larger than that of the *small threshold*, the maximum difference is only 1.7% (see Table 6). Therefore, for the instances with a large number of customers, it is better to use the *large threshold* to save the computational time; while for the instances with a small number of customers, it is better to use the *small threshold* to obtain a slightly better solution. Moreover, because the result is not sensitive to the threshold, the values of β in Eq. (6) can be used for general cases. Thus, there is no case-sensitive empirical parameter in this heuristic.

Tables 7 and 8 give some statistics on the utilization of vehicles and drivers. If once all accommodations of a vehicle are occupied, this trip is called a *fully loaded trip*. It is noticed from these statistics that only a small portion of trips are fully

Table 4
Computational results of large scale instances

Initial solution	Threshold		Time window	Total number of customers		
				500	1000	2000
CFSS	Small	Maximum CPU	Narrow	931.0	9249.0	–
		Time (seconds)	Wide	1257.0	13,765.0	
		Average CPU	Narrow	748.6	6695.4	
		Time (seconds)	Wide	1010.7	10,042.3	
		Average IG (%)	Narrow	1.0	0.9	
			Wide	1.8	1.7	
	Large	Maximum CPU	Narrow	426.0	2659.0	28,115.0
		Time (seconds)	Wide	448.0	4138.0	**35,706.0**
		Average CPU	Narrow	281.7	2129.1	19,781.7
		Time (seconds)	Wide	338.3	3019.9	27,951.0
		Average IG (%)	Narrow	0.4	0.4	0.3
			Wide	1.2	0.9	0.2
	Small and large	Average DG (%)	Narrow	3.3	3.4	3.5
			Wide	3.2	2.8	4.5
OTOC	Small	Maximum CPU	Narrow	1942.0	16,251.0	–
		Time (seconds)	Wide	2124.0	20,004.0	
		Average CPU	Narrow	1273.9	10,753.3	
		Time (seconds)	Wide	1765.9	16,537.0	
		Average IG (%)	Narrow	0.9	0.6	
			Wide	1.1	0.8	
	Large	Maximum CPU	Narrow	577.0	5651.0	41,061.0
		Time (seconds)	Wide	845.0	7905.0	66,881.0
		Average CPU	Narrow	439.8	3935.3	32,605.3
		Time (seconds)	Wide	682.2	5802.3	49,823.9
		Average IG (%)	Narrow	0.4	0.4	0.3
			Wide	0.6	0.1	0.4
	Small and large	Average DG (%)	Narrow	3.8	4.2	3.6
			Wide	4.6	3.6	3.7

Table 5
Comparison the average values of initial solutions

Threshold		Time window	Total number of customers						
			50	100	150	200	500	1000	2000
Small	CFSS time/OTOC time	Narrow	**1.000**	0.903	0.842	0.787	0.587	0.623	–
		Wide	0.800	0.855	0.666	0.680	0.572	0.607	
	CFSS cost/OTOC cost	Narrow	0.999	0.995	**0.988**	0.999	1.002	0.997	
		Wide	0.994	1.004	1.002	1.010	0.999	0.991	
Large	CFSS time/OTOC time	Narrow	0.500	0.714	0.838	0.703	0.641	0.541	0.607
		Wide	0.833	0.730	0.821	0.821	**0.496**	0.520	0.561
	CFSS cost/OTOC cost	Narrow	1.005	1.000	0.999	1.003	1.003	1.001	0.995
		Wide	0.992	**1.010**	1.004	0.999	1.000	0.992	**0.988**

Table 6
Comparison the average values of the small threshold (ST) and the large threshold (LT)

Initial solution		Time window	Total number of customers					
			50	100	150	200	500	1000
CFSS	ST time/LT time	Narrow	**3.500**	2.600	2.627	2.961	2.657	3.145
		Wide	1.600	2.630	2.380	2.418	2.988	3.325
	ST cost/LT cost	Narrow	0.988	0.989	0.989	0.990	0.994	0.994
		Wide	0.989	0.986	0.988	0.995	0.993	0.992
OTOC	ST time/LT time	Narrow	1.750	2.057	2.616	2.645	2.897	2.733
		Wide	1.667	2.243	2.934	2.918	2.589	2.850
	ST cost/LT cost	Narrow	0.995	0.995	0.992	0.994	0.995	0.998
		wide	0.986	0.992	0.990	**0.983**	0.994	0.993

Table 7
Average numbers of customers, vehicles, drivers and trips for narrow time window instances

			Total number of customers						
			50	100	150	200	500	1000	2000
Total customers		Type I	23.5	47.7	75.4	102.6	252.1	491.4	995.3
		Type II	13.2	25.9	37.2	46.6	125.2	259.5	494.4
		Type III	13.3	26.4	37.4	50.8	122.7	249.1	510.3
CFSS and small threshold	Vehicles/drivers	Type I	3.6/3.8	4.9/5.6	7.4/8.0	9.6/10.7	15.0/17.1	22.9/25.9	–
		Type II	2.3/2.3	3.4/3.6	3.8/4.0	4.3/5.0	9.3/10.5	16.8/18.9	
		Type III	5.3/5.5	9.6/10.5	13.3/14.2	16.0/7.6	37.2/41.2	66.9/75.0	
	Trips		13.2	22.2	30.4	39.4	81.7	142.1	
	Fully-loaded trips		0.0	0.0	0.0	0.0	0.0	0.9	
CFSS and large threshold	Vehicles/drivers	Type I	3.6/3.8	5.0/5.5	7.6/8.7	9.1/10.2	15.1/16.9	22.7/25.8	38.7/43.9
		Type II	2.3/2.4	3.8/4.2	4.1/4.2	4.7/5.2	9.9/11.6	18.8/21.4	28.8/33.0
		Type III	5.5/5.8	9.7/10.6	13.4/14.4	16.8/18.3	36.4/40.5	67.1/74.9	122.7/137.8
	Trips		13.2	22.3	30.6	39.2	81.8	143.5	260.4
	Fully-loaded trips		0.0	0.0	0.0	0.0	0.1	0.4	3.7
OTOC and small threshold	Vehicles/drivers	Type I	3.4/3.6	5.3/5.8	7.2/7.6	8.6/9.5	15.0/16.0	23.3/26.7	–
		Type II	2.3/2.5	3.5/3.6	4.7/4.9	4.2/4.9	10.0/11.0	16.2/18.4	
		Type III	5.3/5.7	9.6/10.3	12.6/13.9	17.0/18.3	36.0/39.0	68.0/75.4	
	Trips		13.2	22.2	30.7	38.8	81.6	142.2	
	Fully-loaded trips		0.0	0.0	0.0	0.0	0.0	0.6	
OTOC and large threshold	Vehicles/drivers	Type I	3.4/3.6	5.3/5.8	7.7/8.2	8.5/9.8	15.1/16.6	23.5/26.8	38.8/44.4
		Type II	2.1/2.1	3.3/3.4	4.5/4.7	4.7/5.2	8.9/10.5	15.9/18.3	28.7/32.9
		Type III	5.7/5.9	9.7/10.6	12.4/13.7	16.5/17.8	36.9/41.0	68.8/76.5	123.7/139.1
	Trips		13.1	22.4	31.1	39.3	81.4	142.5	262.5
	Fully-loaded trips		0.0	0.0	0.0	0.0	0.0	0.5	4.2

Table 8
Average numbers of customers, vehicles, drivers and trips for wide time window instances

			Total number of customers						
			50	100	150	200	500	1000	2000
Total customers		Type I	25.0	50.9	74.7	103.3	245.1	499.9	993.1
		Type II	12.0	24.2	37.4	48.5	128.5	240.6	502.8
		Type III	13.0	24.9	37.9	48.2	126.4	259.5	504.1
CFSS and small threshold	Vehicles/drivers	Type I	3.0/3.1	4.5/5.2	5.5/6.1	7.1/8.0	10.8/11.9	18.1/20.6	–
		Type II	1.2/1.2	2.1/2.3	3.1/3.4	3.4/4.0	9.7/10.6	13.5/15.2	
		Type III	5.6/5.9	9.0/9.7	12.8/13.4	15.7/16.9	34.4/37.6	65.1/72.6	
	Trips		11.1	19.4	24.8	32.8	68.4	122.3	
	Fully-loaded trips		0.1	0.3	0.3	0.4	5.2	23.7	
CFSS and large threshold	Vehicles/drivers	Type I	3.0/3.2	4.2/4.8	5.5/5.9	6.9/7.7	10.6/11.4	17.7/20.1	25.4/28.7
		Type II	1.5/1.5	2.9/2.9	3.1/3.4	3.9/4.5	9.2/10.0	14.6/17.0	28.3/31.1
		Type III	5.4/5.8	9.6/10.0	13.2/13.8	15.9/17.2	35.0/38.7	63.6/69.5	116.5/129.4
	Trips		11.3	19.5	25.3	33.1	69.1	120.8	219.1
	Fully-loaded trips		0.0	0.1	0.5	0.4	6.1	22.8	77.8
OTOC and small threshold	Vehicles/drivers	Type I	2.9/3.0	4.3/4.8	6.1/6.6	6.8/7.2	10.5/11.7	16.4/18.4	–
		Type II	1.9/1.9	2.7/2.8	3.0/3.2	4.6/4.9	9.3/10.3	13.9/15.7	
		Type III	5.6/5.9	9.0/9.5	12.2/13.1	15.1/16.2	33.8/37.7	66.1/72.6	
	Trips		11.5	19.1	25.3	31.6	68.4	122.1	
	Fully-loaded trips		0.0	0.3	0.3	0.4	5.3	20.5	
OTOC and large threshold	Vehicles/drivers	Type I	3.0/3.2	4.5/4.9	6.5/6.8	7.2/7.5	10.1/11.5	15.8/17.5	22.8/26.3
		Type II	1.6/1.6	2.6/2.7	2.8/2.9	4.4/4.5	10.3/11.1	14.7/16.7	28.1/32.4
		Type III	5.6/5.8	9.0/9.4	12.5/13.2	15.5/16.5	33.4/37.6	64.9/70.6	119.6/132.4
	Trips		11.4	19.1	25.2	32.4	67.7	120.8	219.6
	Fully-loaded trips		0.0	0.5	0.5	0.1	5.4	22.7	71.0

loaded. This implies that the vehicle capacity constraint is less important than other constraints. It could be a waste if using vehicles with many accommodations. It is also observed that the average number of drivers is greater than the average number of vehicles. This is because there is a break time constraint imposed on drivers. Moreover, these statistics show that the least cost schedule does not definitely have the least number of average trips. This is because different types of vehicles and drivers have different rates and the final cost is determined by the combination of these vehicles and drivers.

Usually the computational time and the gap of wide time window instances are greater than those of narrow time window instances. This is because the local search neighbourhood is larger if the time window is wider. Moreover, it is observed from Tables 7 and 8 that wider time window instances requires fewer number of trips, vehicles and drivers. Therefore, the agency may prefer the wide time window especially when the resources are limited. However, the customer usually prefers the narrow time window. Therefore, a negotiation between the agency and the customer is needed.

5. Conclusions

This paper presents a fast heuristic, which is dedicated to solving the large-scale static DARP under complex constraints. Intensive numerical experiments reveal that this heuristic is capable of quickly obtaining acceptable results and not sensitive to the initial solution. Moreover, it is flexible to cope with many practical constraints and does not contain any case-sensitive empirical parameter. These good performances are the results of the following techniques:

(1) The network of the problem is reduced by removing inadmissible arcs and some constants are calculated in advance.
(2) A good initial solution can be quickly constructed in a simple way.
(3) Simple elementary local searches are properly organized together. The MT facilitates the insertion operation. The inter-trip exchange and the intra-trip exchange are carried out in parallel.
(4) The properly selected SOF points out a promising direction for the diversification. Then larger neighbourhoods can be efficiently explored by simple local searches. Thus, the POF is improved directly and indirectly. Actually, this idea of diversification can be generalized by iteratively using the third, the forth… objective functions, which can be generally called as auxiliary objective functions (AOF). However, it may be difficult to find many AOF and the trade-off should be drawn between the quality of the solution and the computational time.
(5) By evaluating the AWC, some good trips can be excluded from the fine-tuning. Thus, some useless explorations are avoided and the remaining trips get some chances to be refined further.

Although only the static DARP are solved in this paper, some techniques listed above (such as the AOF and the intensification strategy) can be used to solve other problems.

Acknowledgments

This work is supported by a post-doctoral fellowship granted by the Conseil Regional de Champagne-Ardenne. This support is gratefully acknowledged. Thanks also go to anonymous referees. This paper has been greatly improved according to their valuable comments.

Appendix A. The modified column generation algorithm

Because the problem considered in this paper is a little bit different from the problem in the paper of Dumas et al. (1991), small modifications are needed. The modified CG algorithm is briefly described as follows.

The problem is formulated as:

$$\min \quad \sum_{j\in\Omega} c_j X_j \tag{A1}$$

$$\text{s.t.} \quad \sum_{j\in\Omega} a_{ij} X_j = 1, \quad i = 1,2,\ldots,n, \tag{A2}$$

$$\sum_{j\in\Omega} v_{kj} X_j \leqslant V_k, \quad k = \mathrm{I},\mathrm{II},\mathrm{III}, \tag{A3}$$

$$\sum_{j\in\Omega} d_{kj} X_j \leqslant D_k, \quad k = \mathrm{I},\mathrm{II},\mathrm{III}, \tag{A4}$$

$$X_j \in \{0,1\} \quad j \in \Omega, \tag{A5}$$

where c_j is the cost of trip j; Ω, the set of admissible trips; X_j, a binary variable; 1 if trip j is used; 0 otherwise; a_{ij}, a binary constant; 1 if customer i is serviced in trip j; 0 otherwise; n, the number of customers; v_{kj}, a binary constant; 1 if vehicle k is used by trip j; 0 otherwise; V_k, the number of vehicles of type k; d_{kj}, a binary constant; 1 if driver k services trip j; 0 otherwise; D_k, the number of drivers of type k.

Assuming the number of vehicles and drivers are large enough to meet all requests, the lower bound of the solution is obtained by solving the linear relaxation version of the above set partitioning formulation with the CG procedure. By this procedure, the linear relaxation problem is decomposed into a restricted master problem (RMP) and three SPs, one for each combination of the vehicle and the driver. The framework of this CG procedure is presented as follows:

Step 1: Initially, the RMP contains n columns, i.e., one pair of vehicle and driver for each customer request. Denote the maximum number of the newly generated columns as q. Set $q = n/10$.

Step 2: Solve the current RMP to get the dual variables π_i^c $(i = 1, 2, \dots, n)$, π_k^v and π_k^d $(k = \mathrm{I}, \mathrm{II}, \mathrm{III})$, which are corresponding to the constraints (A2)–(A4), respectively. For each SP k, the reduced cost of a column (trip) j is given by:

$$r_j = c_j - \sum_{i \in S_j} \pi_i^c - \pi_k^v - \pi_k^d, \tag{A6}$$

where set S_j contains all the customers in trip j.

Step 3: Solve all SP to find the first q trips with negative reduced cost. If no trip with negative reduced cost is found, stop. Otherwise generate corresponding columns according these trips and add them to the RMP. Update q to $q + n/10$. Go to step 2.

Note: Because the quality of the simply generated initial columns of RMP are not good, the quality of the newly generated columns in the first a few steps are not very good. This is why the value of q is gradually increased step by step.

The RMP is solved by the Simplex solver of IBM OSL academic package 3.0, which greatly facilitates our programming task. For each SP k, the arc table $\boldsymbol{A}$ (see Section 3.1) is further modified by removing the customers which cannot be serviced by the k type of vehicle. Then the SP is solved by the modified DP algorithm. Follow the idea of Dumas et al. (1991), this algorithm also uses labels to solve the shortest path problem. For the jth partial path P_l^j from the depot to the vertex l, the corresponding label L_l^j is defined as

$$L_l^j \equiv (BT_l^j, PBT_l^j, DT_l^j, PDT_l^j, \boldsymbol{Q}_l^j, RC_l^j, PRC_l^j), \tag{A7}$$

where BT_l^j is the time of beginning the service at vertex l on partial path j in the worst case (when $a_{1\mathrm{st}} = e_{1\mathrm{st}}$); PBT_l^j, the potential time of beginning the service at vertex l on partial path j in the best case (when $a_{1\mathrm{st}} = e_{1\mathrm{st}} + pm_{1\mathrm{st}}$); DT_l^j, the duration of partial path j in the worst case (when $a_{1\mathrm{st}} = e_{1\mathrm{st}}$); PDT_l^j, the potential duration of partial path j in the best case (when $a_{1\mathrm{st}} = e_{1\mathrm{st}} + pm_{1\mathrm{st}}$); $\boldsymbol{Q}_l^j$, the set of all on board customers on partial path j; RC_l^j, the reduced cost of partial path j in the worst case (when $a_{1\mathrm{st}} = e_{1\mathrm{st}}$); PRC_l^j, the potential reduced cost of partial path j in the best case (when $a_{1\mathrm{st}} = e_{1\mathrm{st}} + pm_{1\mathrm{st}}$). $pm_{1\mathrm{st}}$, the pseudo-MT of the first pickup site on the partial path, which is calculated from the last vertex of the partial path.

The starting time of a trip is determined by the MT, which cannot be determined unless the whole trip has been constructed. Therefore, it is impossible to get the exact information of a partial trip. However, the information in the best case (when $a_{1\mathrm{st}} = e_{1\mathrm{st}} + pm_{1\mathrm{st}}$) can be estimated by the pseudo-MT. Because the pseudo-MT is an upper bound of the actual MT, the information of the best case is called the potential information.

For each label, the following *post-feasibility* criteria are examined:

Criterion 1. A label $(BT_l^j, PBT_l^j, DT_l^j, PDT_l^j, \boldsymbol{Q}_l^j, RC_l^j, PRC_l^j)$ such that $\boldsymbol{Q}_l^j \notin \phi$ and $i \in \boldsymbol{Q}_l^j$ is eliminated if the path extension $v_l \to v_{n+i} \to v_0$ is infeasible.

Criterion 2. A label $(BT_l^j, PBT_l^j, DT_l^j, PDT_l^j, \boldsymbol{Q}_l^j, RC_l^j, PRC_l^j)$ such that $\boldsymbol{Q}_l^j \notin \phi$ and $i1, i2 \in \boldsymbol{Q}_l^j$ is eliminated if both path extensions $v_l \to v_{n+i1} \to v_{n+i2} \to v_0$ and $v_l \to v_{n+i2} \to v_{n+i1} \to v_0$ are infeasible.

The above two criteria are almost the same as the Proposition 1 and the Proposition 2 in the paper of Dumas et al. (1991). However, when examining the feasibility of the extension, the MT and the maximum driving duration are considered. Moreover, if v_l is the pickup site and has already been serviced in the partial trip, the extension to v_l will not be considered.

Criterion 3. If two labels are such that $\boldsymbol{Q}_l^j \subseteq \boldsymbol{Q}_l^{j'}$, $DT_l^j \leqslant PDT_l^{j'}$, $BT_l^j \leqslant PBT_l^{j'}$, $RC_l^j + (PBT_l^{j'} - BT_l^j) \times$ waiting rate $\leqslant PRC_l^{j'}$ and the on board customer in trip j takes the same or lower level of the accommodation than the same customer in trip j', the label $L_l^{j'} = (BT_l^{j'}, PBT_l^{j'}, DT_l^{j'}, PDT_l^{j'}, \boldsymbol{Q}_l^{j'}, C_l^{j'}, PC_l^{j'})$ can be eliminated.

The above criterion is proposed based on the fact that if any feasible path extension of $P_l^{j'}$ in the best case is also feasible for the path P_l^j in the worst case at lesser cost, the label $L_l^{j'}$ can be eliminated.

Because $PDT_l^j \leqslant DT_l^j, DT_l^j \leqslant PDT_l^{j'}$ ensures that if the extension of the partial path $P_l^{j'}$ will not violate the maximum driving duration constraint, the same extension of partial path P_l^j is also valid.

Because $PBT_l^{j'} \leqslant BT_l^{j'}$ and the earlier to start the service, the less possible to violate the time window constraint, $BT_l^j \leqslant PBT_l^{j'}$ ensures that if the extension of the partial path $P_l^{j'}$ will not violate the time window constraint, the same extension of partial path P_l^j is also valid.

Because $PRC_l^j \leqslant RC_l^j$ and the earlier to start the service, the more possible to increase the waiting cost, $RC_l^j + (PBT_l^{j'} - BT_l^j) \times$ waiting rate $\leqslant PRC_l^{j'}$ ensures that the worst cost of the extension of the partial path P_l^j is less than the best cost of the extension of the partial path $P_l^{j'}$.

Because the matching of customers and accommodations is considered here, it must ensure that if the extension of the partial path $P_l^{j'}$ will not violate the vehicle capacity constraint, the same extension of partial path P_l^j is also valid.

Because the reduced cost satisfies the triangle inequality, the on board customers on partial path $P_l^{j'}$ can contain the customer who are not on board in partial path P_l^j, i.e., $\boldsymbol{Q}_l^j \subseteq \boldsymbol{Q}_l^{j'}$. This has been proved in the paper of Dumas et al. (1991).

Because we use the MT to determine the starting time of the trip and consider dominance criterion in the worst case, the modified CG program greatly slows down. However, we just need some lower bounds of the instances with small number of customers to compare with the results from the heuristic. Here the accuracy of the result is most interested.

References

Backer, B., Furnon, V., Shaw, P., Kilby, P., Prosser, P., 2000. Solving vehicle routing problem using constraint programming and metaheuristics. Journal of Heuristics 6, 501–523.

Bodin, L.D., Sexton, T., 1986. The multi-vehicle subscriber dial-a-ride problem. TIMS Studies in Management Science 26, 73–86.

Cordeau, J.F., Laporte G., 2002. The dial-a-ride problem: Variants, modeling issues and algorithms, Technical report ISSN: 0711-2440, Les Cahiers du GERAD, Montréal.

Cordeau, J.F., Laporte, G., 2003. A Tabu search heuristic for the static multi-vehicle dial-a-ride problem. Transportation Research Part B 37, 579–594.

Cordone, R., Wolfler Calvo, R., 2001. A heuristic for the vehicle routing problem with time windows. Journal of Heuristics 7, 107–129.

Desaulniers, G., Desrosiers, J., Erdmann, A., Solomon, M.M., Soumis, F., 2002. VRP with pickup and delivery. In: Toth, P., Vigo, D. (Eds), The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, pp. 225–242.

Desrosiers, J., Dumas, Y., Soumis, F., 1986. A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows. American Journal of Mathematical and Management Sciences 6, 301–325.

Desrosiers, J., Dumas, Y., Soumis, F., Taillefer, S., Villeneuve, D., 1991. An algorithm for mini-clustering in handicapped transport. Cahier du GERAD G-91-22, Ecole des Hautes Etudes commerciales, Montréal.

Dorigo, M., Maniezzo, V., Colorni, A., 1991. The ant system: An autocatalytic optimizing process. Technical report No. 91-016 revised. Politecnico di Milano, Italy.

Dumas, Y., Desrosiers, J., Soumis, F., 1989. Large scale multi-vehicle dial-a-ride problems. Cahier du GERAD G-89-30, Ecole des Hautes Etudes commerciales, Montréal.

Dumas, Y., Desrosiers, J., Soumis, F., 1991. The pickup and delivery problem with time windows. European Journal of Operational Research 54, 7–22.

Gillett, B., Miller, L., 1974. A heuristic algorithm for the vehicle dispatch problem. Operations Research 22, 340–349.

Glover, F., Laguna, M., 1997. Tabu Search. Kluwer, Boston.

Healy, P., Moll, R., 1995. A new extension of local search applied to the dial-a-ride problem. European Journal of Operational Research 83, 83–104.

Holland, J.H., 1975. Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor.

Hunsaker, B., Savelsbergh, M., 2002. Efficient feasibility testing for dial-a-ride problems. Operations Research Letters 30, 169–173.

Ioachim, I., Desrosiers, J., Dumas, Y., Solomon, M.M., 1995. A request clustering algorithm for door-to-door handicapped transportation. Transportation Science 29, 63–78.

Jaw, J., Odoni, A.R., Psaraftis, H.N., Wilson, N.H.M., 1986. A heuristic algorithm for the multi-vehicle advance-request dial-a-ride problem with time windows. Transportation Research B 20, 243–257.

Kinderwater, G.A.P., Savelsbergh, M.W.P., 1997. Vehicle routing: Handling edge exchanges. In: Aarts, E.H.L., Lenstra, J.K. (Eds.), Local Search in Combinatorial Optimization. Wiley, Chichester, pp. 337–360.

Kirckpatrick, S., Gelatt Jr., C., Vecchi, M., 1983. Optimization by simulated annealing. Science 220, 671–680.

Laporte, G., Gendreau, M., Potvin, J.Y., Semet, F., 2000. Classical and modern heuristics for the vehicle routing problem. International Transactions in Operational Research 7, 285–300.

Lin, S., 1965. Computer solutions of the travelling salesman problem. Bell System Technical Journal 44, 2245–2269.

Lin, S., Kernighan, B., 1973. An effective heuristic algorithm for the travelling salesman problem. Operations Research 21, 498–516.

Or, I., 1976. Travelling salesman-type combinatorial optimization problems and their relation to the logistics of regional blood banking. Ph.D. dissertation, Northwestern University, Evanston, IL.

Nanry, W.P., Barnes, J.W., 2000. Solving the pickup and delivery problem with time windows using reactive tabu search. Transportation Research B 34, 107–121.

Psaraftis, H.N., 1980. A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. Transportation Science 14, 130–154.

Psaraftis, H.N., 1983a. An exact algorithm for the single vehicle many-to-many immediate request dial-a-ride problem with time windows. Transportation Science 17, 351–357.

Psaraftis, H.N., 1983b. *k*-Interchange procedures for local search in a precedence-constrained routing problem. European Journal of Operational Research 13, 391–402.

Roy, S., Rousseau, J.M., Lapalme, G., Ferland, J.A., 1984a. Routing and scheduling for the transportation of disabled persons—the algorithm. Technical Report TP 5596E, Centre de Recherche sur les Transports, Montréal.

Roy, S., Rousseau, J.M., Lapalme, G., Ferland, J.A., 1984b. Routing and scheduling for the transportation of disabled persons—the tests. Technical Report TP 5598E, Centre de Recherche sur les Transports, Montréal.

Savelsbergh, M.W.P., 1992. The vehicle routing problem with time windows: Minimizing route duration. ORSA Journal on Computing 4, 146–154.

Savelsbergh, M.W.P., Sol, M., 1998. Drive: Dynamic routing of independent vehicles. Operations Research 46, 474–490.

Tan, K.C., Lee, L.H., Ou, K., 2001. Artificial intelligence heuristics in solving vehicle routing problems with time window constraints. Engineering applications of artificial intelligence 14, 825–837.

Toth, P., Vigo, D., 1997. Heuristic algorithms for the handicapped persons transportation problem. Transportation Science 31, 60–71.

Van Breedam, A., 2001. Comparing descent heuristics and metaheuristics for the vehicle routing problem. Computers and Operations Research 28, 289–315.

Voudouris, C., Tsang, E., 1995. Guided local search. Technical Report CSM-247, Department of Computer Science, University of Essex, Colchester, UK.

Wilson, H., Sussman, J., Higonnet, B., 1971. Scheduling algorithms for dial-a-ride systems. Technical Report USL TR-70-13, Urban Systems Laboratory, MIT, Boston.

Xu, H., Chen, Z.L., Rajagopal, S., Arunapuram, S., 2003. Solving a practical pickup and delivery problem. Transportation Science 37, 347–367.