Available online at www.sciencedirect.com
ScienceDirect
Computers & Operations Research 34 (2007) 1008–1032
computers & operations research
www.elsevier.com/locate/cor

A decision support system for the single-depot vehicle rescheduling problem

Jing-Quan Li[a], Denis Borenstein[b,*], Pitu B. Mirchandani[a]

[a] *Systems and Industrial Engineering, The University of Arizona, Tucson, AZ 85721, USA*
[b] *Management School, Universidade Federal do Rio Grande do Sul, R. Washington Luis 855, Porto Alegre 90010-460, RS, Brazil*

Available online 22 December 2005

Abstract

Disruptions in trips can prevent vehicles from executing their schedules as planned. Mechanical failures, accidents, and traffic congestion often hinder a vehicle schedule. When a vehicle on a scheduled trip breaks down, one or more vehicles need to be rescheduled to serve the passengers/cargo (if there are any) on that trip. The main objective of the *vehicle rescheduling problem* (VRSP) is to minimize operation and delay costs, while serving the passengers/cargo on the disrupted trip and completing all remaining trips that include the disrupted one. We report on a prototype decision support system (DSS) that recommends solutions for the single-depot rescheduling as well as *vehicle scheduling* (VSP) problems, since VRSP is closely related to VSP. The system was designed for human schedulers to obtain optimal vehicle assignments and reassignments. An experimental study, using randomly generated data, shows the efficiency of the developed algorithm. A real world problem, which involves the solid waste collection operational planning for a Brazilian city, is selected as the case study to illustrate the effectiveness of the developed DSS.

Keywords: Vehicle scheduling; Decision support systems; Operational planning; Rescheduling

* Corresponding author. Tel.: +55 51 316 3459; fax: +55 51 316 3991.
E-mail addresses: jingquan@email.arizona.edu (J.-Q. Li), denisb@ea.ufrgs.br, denis@adm.ufrgs.br (D. Borenstein), pitu@sie.arizona.edu (P.B. Mirchandani).

0305-0548/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2005.05.022

1. Introduction

Transportation and logistic systems often encounter disruptions that prevent them from operating as planned. Severe weather conditions, accidents, and the breakdown of vehicles are examples of possible disruptions that demand the rescheduling of vehicles. The vehicle rescheduling problem (VRSP) consists of defining a new schedule for a set of previously scheduled trips, given that a trip has been severely disrupted. The main objective of the vehicle rescheduling problem is to minimize the involved operation and delay costs, under the condition that uncompleted trips must be finished (including the disrupted one).

Rescheduling is usually made by human experts. They often employ common sense and past experiences that are blended in a fuzzy, sometimes inconsistent, and not well-understood way. When fleet size is limited and disruptions are frequent [1], good automated rescheduling tools with effective algorithms become important for the following reasons:

- For large problems that involve thousands of trips and hundreds of vehicles, the number of feasible solutions for the rescheduling problem may be so large that even the most skillful human scheduler may overlook good rescheduling options. The availability of a systematic procedure, based on an optimization algorithm, will guarantee that a good reassignment is obtained.
- Rescheduling needs to be obtained quickly, since disruption delays will have a negative influence on system performance, because they simultaneously result in increasing costs and deterioration of the level of service.
- Since crews might be reassigned to a new schedule, the computation of VRSP needs to be completed as fast as possible. Without automated rescheduling tools, often the easiest and quickest approach of the human scheduler is to send a vehicle from the depot to backup the disrupted trip, which might be a poor solution.

This means that there is a potential for improvement through a decision support system (DSS) due to the complexity of the problem. The development of new information technologies (e.g., global positioning system, geographical information systems, cellular phones, etc.) has raised the interest in automatic recovery strategies. As real-time information is now available at low-cost, human schedulers are forced to react to unexpected events in real time [2]. The VRSP arises in a wide array of practical applications such as school bus routing, operational planning of public transportation systems, industrial/hospital refuse collection, mail delivery, telecommunication systems, etc.

The VRSP can be approached as a dynamic version of the classical vehicle scheduling problem where assignments are generated dynamically. Although there is an abundant literature dealing with vehicle scheduling (usually in the context of planning a schedule), there is still a lack of methodologies that can efficiently solve the VRSP. In this paper, we develop the specifications for of a prototype decision support system for both the single-depot VSP (SDVSP) and single-depot VRSP (SDVRSP) problems. The DSS is designed to help human schedulers to assign and reassign vehicle schedules towards obtaining optimal solution for both problems. The robust capability of solving both problems, with emphasis on the automatic rescheduling through integration with real time information, makes the DSS an effective computer-based tool to be used by human schedulers in transportation/logistic companies. The main contribution of this paper is to describe a DSS, which is able to support both SDVSP and SDVRSP in an efficient and effective manner.

The paper is organized as follows. Section 2 reviews the literature on the static and dynamic contexts of the VSP. The descriptions of the SDVSP and SDVRSP problems and models are provided in Section 3. Section 4 describes the developed DSS, with a discussion about the procedures used to optimize both the SDVSP and the SDVRSP. Section 5 presents computational experiments to evaluate the performance of the algorithms by comparing the developed algorithm with CPLEX, using randomly generated data. In Section 6, the application of the DSS for a real-world problem, the solid waste collection operational planning for a Brazilian city, is described. In the concluding Section 7, summary of the results and areas of future research are discussed.

2. Literature review

In this section we present an overview of the static and dynamic SDVSP literature. As a classical optimization problem [3], the static SDVSP can be defined as the problem of assigning vehicles to a set of predetermined trips with prescribed starting and ending times while minimizing the total capital and travel costs, and satisfying the following constraints: (i) every trip has to be assigned to exactly one vehicle; (ii) each vehicle performs a feasible sequence of trips; and (iii) only one depot exists. Deterministic and static versions, where all the characteristics of the trips are known in advance and every parameter assumed certain, have been widely studied in the literature. Overviews of algorithms and applications for the static SDVSP and some of its extensions can be found in [3–6]. This problem has already been formulated as a linear assignment problem, a transportation problem, a minimum-cost flow problem, a quasi-assignment problem, and a matching problem in the literature.

Bokinge and Hasselstrom [7] propose a minimum-cost flow approach that uses a significant reduction on the size of the model in terms of the number of variables at the price of an increased number of constraints. A successive shortest-path algorithm and variations for the SDVSP were proposed in [8–10].

Paixão and Branco [11] propose an $O(n^3)$ quasi-assignment algorithm that is especially designed for the SDVSP. Freling et al. [6] use a quasi-assignment model and employ a forward/reverse auction algorithm for the solution. Computational results show that the approach relating to quasi-assignment significantly outperforms approaches based on the minimum-cost flow and linear-assignment models.

Currently, one of the best model and algorithm for SDVSP is the quasi-assignment and auction algorithm [6], respectively. Bertsekas and Eckstein [12] show that if ε-scaling is used, that is applying the auction algorithm starting with a large value of ε and gradually reducing it to a final value that is less than $1/n$, the complexity is $O(nm \log nC)$, where n is the number of elements to assign, m is the number of possible assignments between pairs of elements, and C is the maximum of absolute benefits.

Since automatic recovery from disruptions is a relatively new operational strategy, the literature related to the dynamic aspects of vehicle rescheduling is sparse. The majority of the research conducted in this area is related with the dynamic vehicle routing problem (VRP) and with the airline recovery problem.

Dynamic vehicle routing problem, where new requests arise in midst of operations, has gained increasing attention since the late eighties [2]. Recent surveys on dynamic VRP can be found in [13–15]. Ichoua et al. [2] have designed a heuristic based on tabu search to handle the real-time vehicle dispatching problem. Yang et al. [16] have developed a general framework for dynamic vehicle routing problem in which new requests can arrive during operations, where the travel time is considered as a variable. Stochastic programming has also been used to tackle uncertainty in vehicle scheduling and routing (e.g., see [17,18]).

Although the above efforts have studied dynamic aspects of the vehicle scheduling/routing, the requirements of the vehicle rescheduling problem, including finding a backup vehicle and picking up passengers/cargo from the breakdown vehicle, have not been addressed.

The literature on aircraft disruption includes [19–22]. These works developed optimization models that reschedule legs and reroute aircraft by minimizing rerouting and cancellation costs. The developed models are airline specific and involve a limited number of trips and airplanes; and most of associated algorithms are computationally intensive. As a consequence, they cannot be used for a more generic problem such as the VRSP, which can involve thousands of trips and hundreds of vehicles.

To the best of our knowledge, the only contributions towards solving our dynamic VSP are found in [23,24]. Huisman et al. [23] have proposed an approach to the dynamic VSP by solving a sequence of optimization problems. Their work is motivated by designing robust vehicle schedules to avoid trips starting late in environments characterized by significant traffic jams. Li et al. [24] have developed a parallel auction algorithm for bus rescheduling. There, the bus rescheduling problem is modeled as several VSP problems, each one corresponding to the use of a different vehicle as an alternative to backup the disrupted trip. All problems are solved using a parallel implementation of a combined forward-backward auction algorithm developed by Freling et al. [6] designed for the quasi-assignment problem. The parallel algorithm has been proven to be computationally efficient, even for large problems. Therefore, it has potential to be used in automated recovery tools.

Given the complexity of the SDVRSP problem, there is a gap between the existence of efficient algorithms and their application to real world cases [25]. For an effective application of these algorithms, a great deal of knowledge and expertise is required. Furthermore, implementation of the developed algorithms is not so straightforward, after the specific requirements of transportation and logistics companies are considered. As a result, very few companies utilize automated rescheduling policies. Thus, the development of a computer-based tool, which is capable of providing a friendly environment for users and emphasizing flexibility, efficiency, and adaptability, becomes an important aspect for the effective use of real-time vehicle scheduling strategies. Although some DSSs have already been developed in the area of vehicle routing [26–28] and scheduling [3], they do not address the vehicle rescheduling problem.

3. Problem description

Vehicle rescheduling problems arise when a scheduled trip is severely disrupted. Although most disruptions do not warrant rescheduling to be performed, some disruptions are severe enough to prevent the trip being finished in a reasonable amount of time. Vehicle breakdown and accidents are examples of severe disruptions that might request a reassignment of vehicles, mainly if the vehicle of the disrupted trip is carrying passengers or cargo. Fig. 1 describes a typical decision process involving both the scheduling and the disruption decision-making sub-processes. The scheduling decision-making process can be characterized as the classical single-depot vehicle scheduling problem.

Before considering mathematical formulations, we introduce some definitions and notations. *Deadheading trips* are movements of vehicles without serving passengers. Trips i and j constitute *compatible pair of trips* if the same vehicle can reach the starting point of trip j after it finishes the trip i. We use a binary relation symbol ct to determine whether two trips characterize a compatible pair of trips. If $ct(i, j) = 1$, trips i and j constitute a compatible pair of trips. Otherwise, if $ct(i, j) = 0$, trips i and j do not constitute a compatible pair of trips.

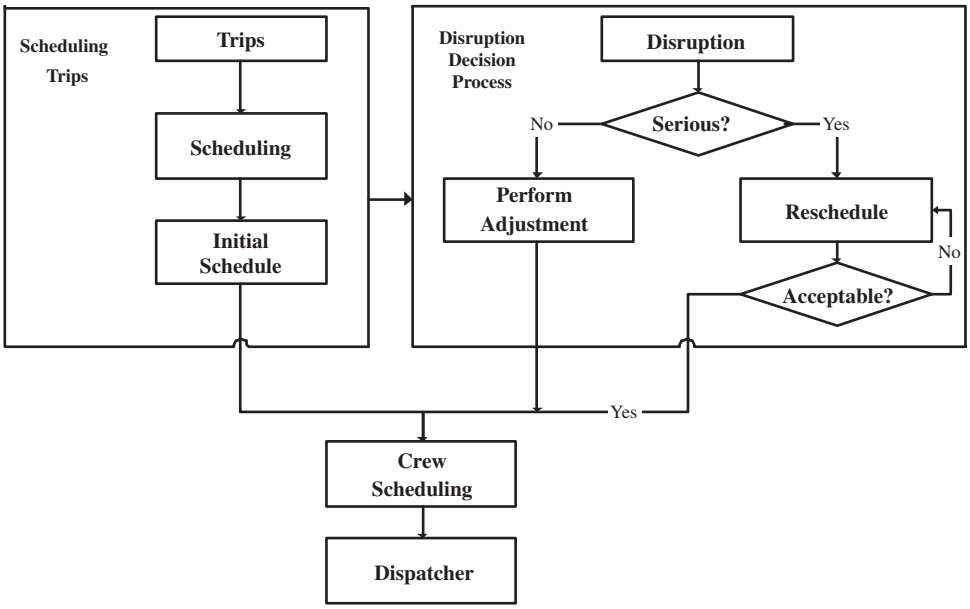


Fig. 1. Scheduling and disruption decision-making process.

Let $N = \{1, 2, \ldots, n\}$ be the set of trips, numbered according to increasing starting time, and let $E = \{(i, j) | i < j, ct(i, j) = 1, i \in N, j \in N\}$ be the set of arcs corresponding to deadheading trips. The vehicle-scheduling network be $G = \langle V, Z \rangle$ with nodes $V = N \cup s, t$ and arcs $Z = E \cup (s \times N) \cup (t \times N)$, where s and t denote the same depot in the network, with s simply meaning the depot as a starting point, and t as the terminating point. A path from s to t in the network represents a feasible vehicle schedule. The SDVSP can be formulated as a quasi-assignment problem as follows [6]:

$$\min \quad \sum_{(i,j)\in Z} c_{ij} y_{ij}$$

s.t.

$$\sum_{j:(i,j)\in Z} y_{ij} = 1 \quad \forall i \in N,$$

$$\sum_{i:(i,j)\in Z} y_{ij} = 1 \quad \forall j \in N,$$

$$y_{ij} \in \{0, 1\} \quad \forall (i, j) \in Z,$$

where

$c_{ij} =$ cost of arc $(i, j) \in Z$,

$$y_{ij} = \begin{cases} 1 & \text{if a vehicle is assigned to trip } j \text{ directly after trip } i, \\ 0 & \text{otherwise.} \end{cases}$$

The constraints in the formulation assure that each trip is assigned to exactly one predecessor and one successor. We refer to Freling et al. [6] for a discussion about algorithms to solve the SDVSP.

If a disruption occurs, the human schedulers use their expertise to define if the disruption is severe or not. If it is not severe, only small adjustments are necessary and, in general, the initial schedule is not changed. In case of a serious disruption, rescheduling is necessary, and the initial schedule (excluding the already finished trips) is used as a basis for rescheduling. This paper presents in Section 3.3 how vehicle rescheduling is formulated, modeled, and optimally solved. However, it is possible that even an optimal solution cannot be implemented in practice due to real-world considerations, such as crew recovery issues, or poor service for important users of the system. In this sense, the human schedulers can use their expertise to find a good compromise between the mathematical solution and these real world issues. Therefore, some schedulers are willing to accept "suitable" solutions after taking into consideration the company strategy to deal with this problem, rather than optimal ones.

It should be noted that the disruption decision-making process is highly dependent on the existence of an effective information system that is able to quickly capture information about disruptions, such as disruption time and place, and the number of passengers or cargo size in the vehicle of the disrupted trip. After the VRSP is solved, the new schedule should also be quickly provided to the involved personnel (vehicle conductors and associated crew). We assume here that the transportation/logistic company has already developed or is able to develop such an information system.

We need to introduce some additional definitions and notations to describe VRSP. To relate to a cut in a graph, we refer to a disrupted trip due to a disabled vehicle, or a vehicle that is effectively inoperable, as a *cut trip*. *Breakdown point* is the point in the cut trip where the trip is disrupted. *Current trip* is the trip on which a vehicle is serving. It includes both regular and deadheading trips. The vehicle that serves the remaining passengers in the cut trip, referred to as the *backup vehicle* in the sequel, can be identified from the trip just served, which will be referred to as *backup trip* in the sequel. The assignment problem is effectively assigning the node representing the backup trip, or assigning (a vehicle from) the depot, to the passengers/cargo of the cut trip. Trip i is an *itinerary compatible trip* with trip j if trip i shares the same itinerary of cut trip j from the breakdown point until its ending point.

The VRSP can be defined as follows. Given a depot and a series of trips with prescribed starting and ending times, given the travel times between all pairs of locations, and given a cut trip, find a feasible minimum-cost reschedule in which (1) each vehicle performs a feasible sequence of trips, and (2) all passengers or cargo (if there are some) on the cut trip are served. There are two possible situations in the SDVRSP. The first one is when the cut trip is a regular one. Unless the disruption is of a nature that it is impossible to reach the disruption point, the passengers/cargo of the cut trip have to be served. The solution comprises of sending a backup vehicle to the breakdown point, and from there completing the cut trip, serving its passengers/cargo. However, since it is highly likely some trips have common itineraries, the passengers/cargo can also be served incidentally by vehicles that cover compatible itineraries after the breakdown point. Let us consider the following situation: a backup vehicle changes its schedule and travels towards the breakdown point, but all the passengers/cargo from the disabled vehicle have been picked up already by vehicles that cover compatible itineraries with the cut trip (see [24] for details). This situation needs to be avoided. The second situation arises when the cut trip is a deadheading trip, the solution is to assign a backup vehicle for the starting location of the next trip of the vehicle on the disrupted deadheading trip. In both cases, it is highly likely that the solution of the SDVRSP provides new routes for a subset of the pre-assigned vehicles. Also, we can expect some delays in the cut trip, mainly in the first situation. The main objective of the SDVRSP is to minimize the total delay cost.

Table 1
Starting and ending times of trips

Trip	Starting time	Ending time
1	5	10
2	1	13
3	20	25
4	22	28

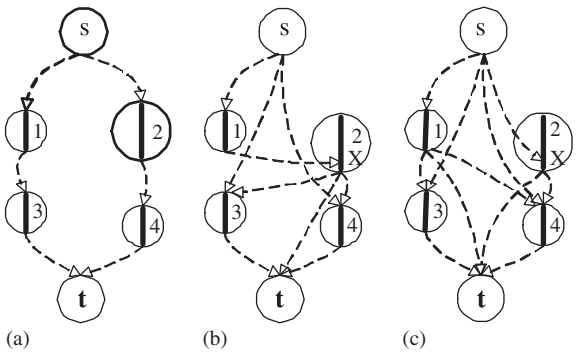


Fig. 2. Example of feasible networks.

The most important aspect of the SDVRSP is that the solution is dependent on the current existing situation and available alternatives to serve the cut trip. Each possible configuration of a recovery can be translated as a corresponding *feasible network*. These feasible networks share the same nodes (i.e., the trips to be served), but have different arcs connecting them. The definition of the set of all possible feasible networks is dependent on the pre-assigned configuration of the trips, the available capacity of the involved vehicles, and times to carry out deadheading and regular trips.

It is possible to have a different feasible network for each possible backup vehicle. We illustrate the idea of feasible networks with an example. Suppose we have to carry out four trips with the travel times indicated in Table 1. Suppose the travel time from the ending point of each trip (or depot) to the starting of a trip is a constant equal to 4 time units. A possible initial scheduling is presented in Fig. 2(a). Each regular trip is a "node" of the feasible network, which is graphically represented as a short line segment to indicate starting and ending points of the trip. Two vehicles can cover the four trips. Vehicle 1 is assigned to trips 1 and 3, while vehicle 2 is assigned to trips 2 and 4.

Suppose vehicle 2 breaks down on trip 2 at time unit 10. There are two possible backup vehicle alternatives. The first is to use vehicle 1; Fig. 2(b) presents the feasible network representing this situation. In this network the dashed lines represent the new possible assignments of vehicles to trips. Observe that an additional vehicle from depot may be used to complete trip 3 or trip 4. The second alternative is to send a new vehicle from depot to the breakdown point in trip 2. This situation is depicted in Fig. 2(c). Both figures present the set of all possible feasible networks, in which a new schedule should be determined. Section 3.2 describes a formal procedure to generate the set of all possible feasible networks.

Based on these feasible networks, we can model the VRSP as a VSP in each feasible network. The VRSP optimal schedule is the one with the minimal cost over the set of all possible feasible networks. It is quite likely that the remaining vehicles have their routes changed to accommodate the disturbances caused by the cut trip. As the number of trips grows, the number of backup vehicle candidates may grow largely. For example, if there are 1300 trips in the network, more than 300 vehicles may be needed to cover these trips [6]. It means that there are about 300 possible backup vehicles, and therefore about 300 possible feasible networks. The large number of backup vehicle candidates makes the problem very difficult to solve. Next section describes in detail the developed DSS for helping human schedulers to solve the decision processes shown in Fig. 1.

4. The decision support system

The proposed prototype of DSS uses the integration of information technology and efficient algorithms to solve both the VSP and the VRSP. Considering the steps presented in Fig. 1, the DSS includes the following requirements:

1. To provide an interactive interface that allows the users to retrieve, build and change trips and vehicles databases.
2. To obtain optimal solutions to the SDVSP.
3. To obtain optimal solution for the SDVRSP when an unexpected event occurs.
4. To provide an interactive environment to create and modify possible operational scenarios.

The DSS is composed of three common basic components: user-interface subsystem, database management subsystem, and model subsystem. Fig. 3 presents the architecture of the DSS. The arrows represent the information flow. Web-based graphic interface is implemented, so that the human schedulers can remotely access the system at any time without the limitations of geographical location and time zone. The DSS is also designed to be independent of the specific platform, and it supports Unix, PC and Apple environments. This allows the user to configure the DSS to meet technical requirements of a specific transportation/logistic company.

4.1. User interface subsystem

This subsystem is a visual interactive tool for scheduling and rescheduling of vehicles. It includes the functions: (i) to control the flow of information among the modules within the software system, (ii) to run the different algorithms within the DSS, (iii) to build, interactively, the databases, and (iv) to present the results. This module uses extensive graphical facilities and menu-driven interfaces to achieve a satisfactory level of interaction, present the output in a meaningful way and provide a smooth and reliable communication with the user.

The user interface provides a meaningful framework within which information can flow in both the directions between the user and the computer, so that the user can take full responsibility on the decision. Fig. 4 presents an example of the several windows in the user interface subsystem. In particular, this window shows the schedules obtained by executing the algorithms in the model subsystem.

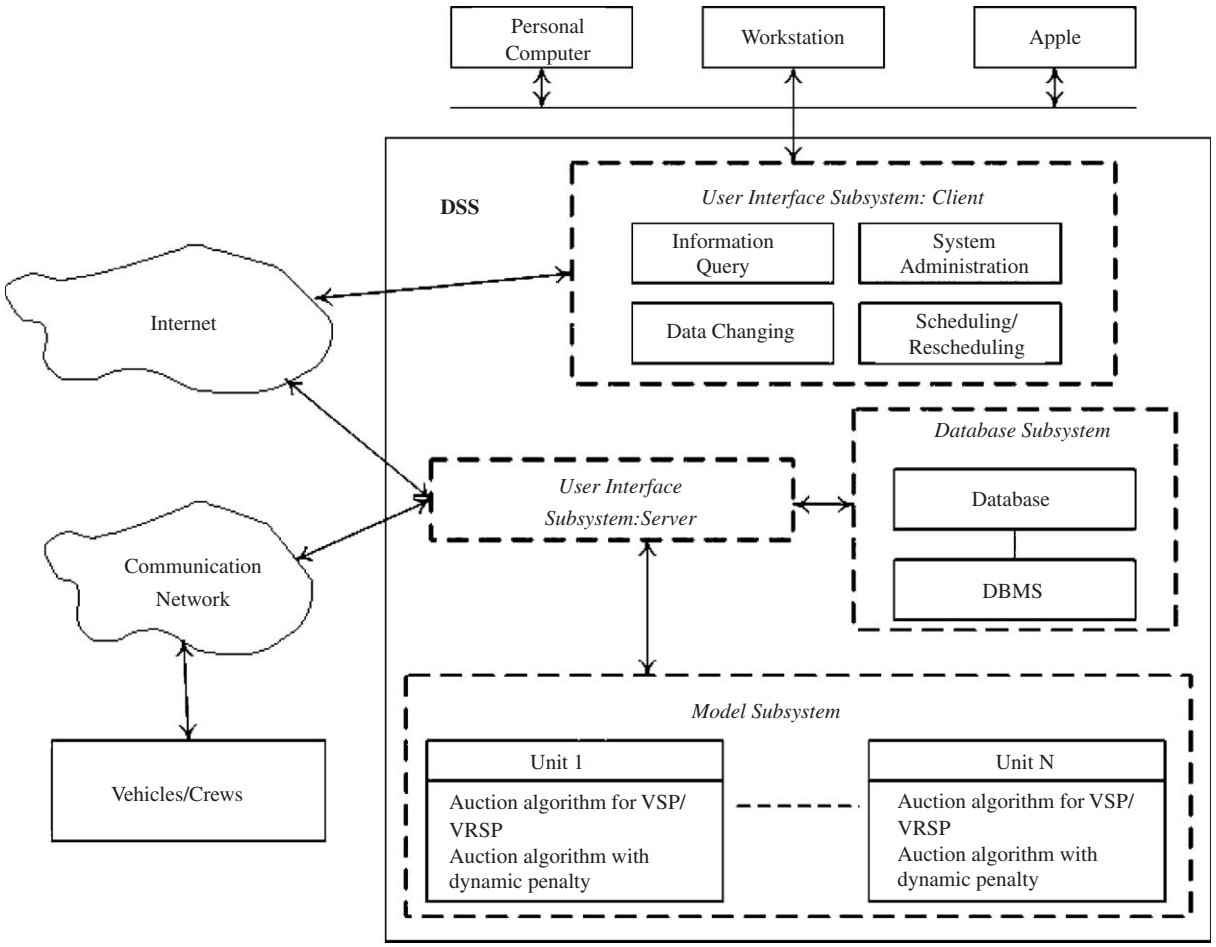


Fig. 3. DSS architecture.

4.2. Database subsystem

The database subsystem is responsible for the management of the databases. The database subsystem contains all the information needed for the scheduling and rescheduling of vehicles. Typical data includes the following attributes: trips (starting and ending times, starting and ending places), vehicles (identity, geographical position, crew and capacity), and routes (vehicle ID and sequence of trips). The database subsystem provides functionality for querying, storing, recovering, and controlling data. Through this module, the company can create, maintain and update all the information available to human schedulers in the "user interface subsystem". Moreover, this subsystem enables companies to obtain/deliver real-time information or data that guarantees the effectiveness of both scheduling and rescheduling decision making processes, automatically updating all changes in attributes during the operation of the system.

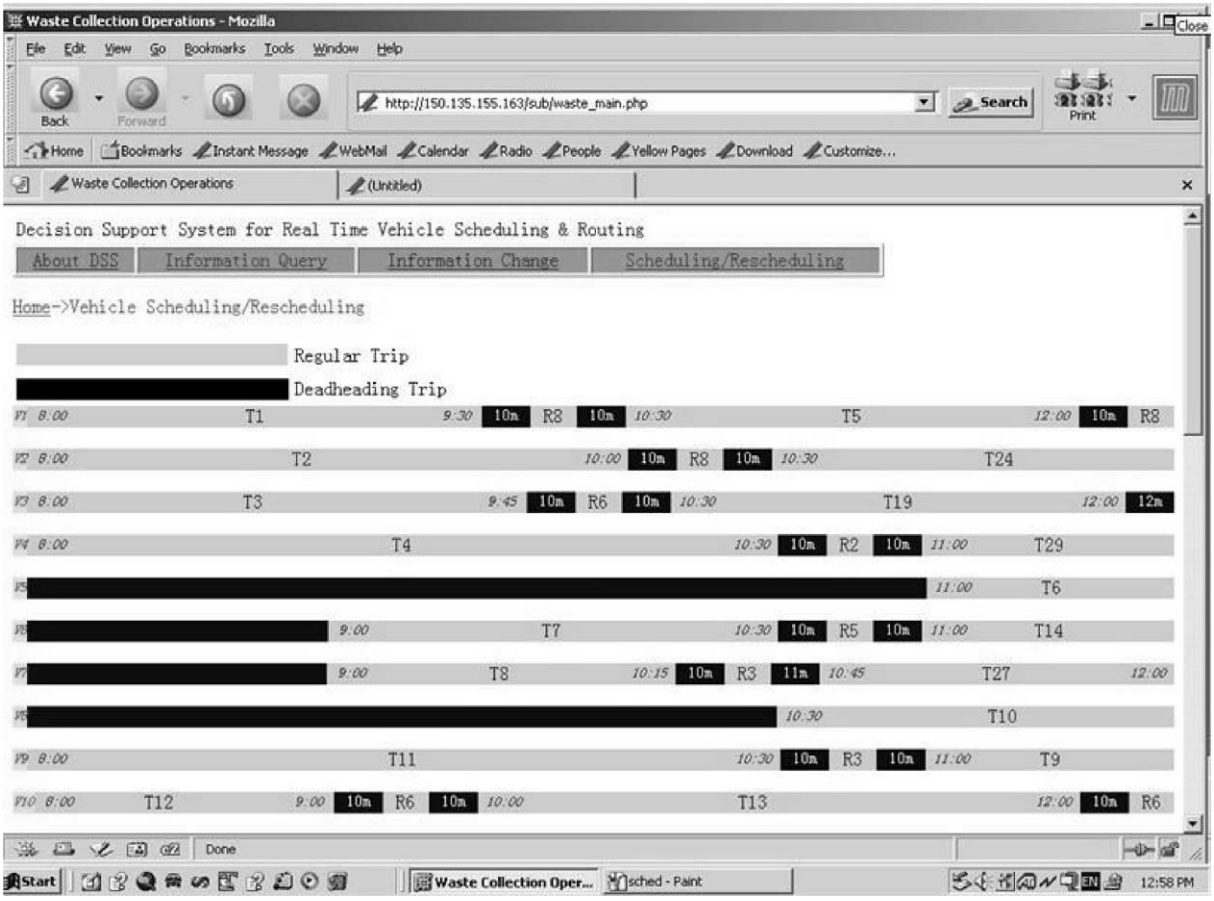


Fig. 4. Graphical representation of schedules in the DSS.

4.3. Model subsystem

The model subsystem is used by the DSS to solve the classical SDVSP and the SDVRSP. Since the former problem is a well-studied topic, this section will focus on the model and the solution approach for the latter problem. Due to their simplicity and efficiency, the quasi-assignment and auction algorithm are selected for solving SDVSP (see Section 2). Since SDVRSP is treated as a sequence of SDVSP problems, the quasi-assignment formulation and the combined forward-backward auction algorithm, developed by [6], are used to model and solve, respectively, the SDVSP within the DSS.

As described in Section 3, the SDVRSP can be modeled as a minimization problem over several SDVSP problems, each one relating to a possible feasible network. It is possible to have a different feasible network for each possible backup vehicle candidate (or each backup trip, since there is a unique correspondence between the vehicle and its current trip in the vehicle scheduling problems). A feasible network is defined formally as follows. Let b denote the cut trip and K be the set of possible backup trip candidates. "Arcs" in the network correspond to vehicle assignment to trips. Let $N' = N - \{b\}$ be the set

of total remaining trips excluding the cut trip, numbered according to the non-decreasing starting times. Define $E(k) = E \cup \{(k, b)\}$. A feasible network for backup trip k can be defined as $G(k) = \langle V, X(k)\rangle$ with nodes $V = N \cup \{s, t\}$ and arcs $X(k) = E(k) \cup (s \times N') \cup (N \times t)$, for $k \in K$, where k is the backup trip. We can define $G = \{G(k)|k \in K\}$ as the set of all feasible networks.

The SDVRSP can be modeled as a quasi-assignment problem, based on the SDVSP formulation presented in Section 3, as follows:

$$\min_{G} \left\{ \min \sum_{(i,j)\in X(k)} c_{ij} y_{ij} + D_k \right\}$$

s.t.

$$\sum_{j:(i,j)\in X(k)} y_{ij} = 1 \quad \forall i \in N,$$

$$\sum_{i:(i,j)\in X(k)} y_{ij} = 1 \quad \forall j \in N,$$

$$y_{ij} \in \{0, 1\} \quad \forall (i, j) \in X(k),$$

where

$c_{ij} =$ cost of arc $(i, j) \in X(k)$,

$D_k =$ delay cost related to the VSP solution for the kth backup trip.

The objective of this formulation is to find a schedule with the minimal operating and delay cost over the set of all possible feasible networks. The constraints in the formulation assure that each trip is assigned to exactly one predecessor and one successor. However, this formulation requires considerable computational time for its solution, since $|G|$ integer linear programs with $|N'|$ zero-one variables need to be solved (see results in Table 2). As $|G|$ and $|N'|$ can be very large, it is necessary to find another practical approach to solve the SDVRSP. Our solution approach is described briefly as follows:

Step 1: Based on the initial schedule, define the set of backup trip (vehicle) candidates.
Step 2: For each backup trip, construct the corresponding feasible network.
Step 3: Solve the VRSP for each feasible network.
Step 4: Select the feasible network with the minimum scheduling cost.

In order to solve the VRSP, we need first to generate the set of all possible feasible networks. This task is based on the available capacity of the involved vehicles, time to reach the breakdown point and the compatibility of itineraries among trips. Actually, the available capacity and travel time are random variables, but in this deterministic model we use average values. Since some trips have common itineraries, the passengers/cargo may be served incidentally by vehicles that cover compatible itineraries after breakdown point. As we mentioned earlier, we need to avoid the situation where a backup vehicle is traveling towards the breakdown point but all the passengers/cargo from the disabled vehicle have been incidentally picked up by the vehicles that cover compatible itineraries with the cut trip.

From the viewpoint of the cut trip, the remaining trips can be divided into two categories: (1) unfinished trips that have compatible itineraries with the cut trip from the breakdown point, and (2) the remaining

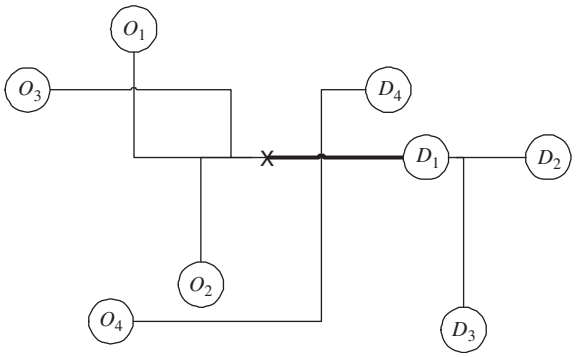


Fig. 5. Example of itinerary compatible trips.

unfinished trips. Fig. 5 illustrates these two categories where paths $O_i - D_i$ denotes trip i from origin O_i to destination D_i. The breakdown point is point X on trip 1. The set of compatible trips with trip 1 from point X is {2, 3}.

Define set A to be the set of unfinished compatible itinerary trips with the cut trip from the point X, ordered by the travel time from their current position to point X. Define set B to be the remaining unfinished trips (including the trips directly from the depot). If the backup trip alternatives are from set A, the backup vehicles can pick up the passengers/cargo incidentally. Although a reschedule may not be necessary, it may be necessary to assign a vehicle from set B to cover unfinished trips originally assigned to the disabled vehicle. If the backup trip alternatives are from set B, backup vehicles need to travel towards the breakdown point for picking up the passengers on the disabled vehicle. The following procedure *Build-Feasible-Networks* was developed to accomplish this task (where K is the set of all possible backup trip candidates):

Procedure Build-Feasible-Networks

Step 1: Divide the unfinished trips in two different sets, A and B. Set $K \leftarrow \emptyset$.

Step 2: Find n^*, the number of backup vehicles serving the trips in $A \neq \emptyset$, using the routine described in Fig. 6, where $C(i)$ be the available capacity of the vehicle serving trip i when it reaches the breakdown point, $T(i)$ be its arrival time at the breakdown point, P be the capacity of the vehicle needed to serve the breakdown trip, T_d be the disruption time, and T_l be a time limit for a backup vehicle to arrive at the breakdown point (only vehicles which arrive at the breakdown point before this limit T_l, are selected as backup vehicle candidates).

Step 3: If the system has a feasible solution $n^* > 0$, and an associated time $T(a_{n^*})$ by which the n^* vehicles serve the passengers/cargo on the disabled vehicle, then we can determine B^*, the set of candidate backup trips from set B, by using

$$B^* = \{m | [T_d \leqslant T(m) < T(a_{n^*}), \forall m \in B, a_i \text{ is the } i\text{th element in set } A(n^*)] \wedge [T(m) \leqslant T_l]\}$$

where $A(n)$ is the subset of A that includes the first n elements of A.

Set $K \leftarrow K \cup B^*$. Go to step 5.

Step 4: If the system of inequalities does not have a feasible solution, set $T(n^*) \leftarrow \infty$, and treat B^* as B. Set $K \leftarrow K \cup B^*$.

```
routine find_n*:
begin
    n* ← 0
    i ← FIRST − ELEMENT(A)
    ReturnValue ← true
    repeat
        if ([C(i) ≤ P] ∧ [T(i) ≥ T_d] ∧ [T(i) ≤ T_l]) then
        begin
            P ← P − C(i)
            n* ← n* + 1
            INSERT(i, K)
        end
        i ← NEXT − ELEMENT(A)
    until ([P ≤ 0] ∨ [i = END(A)])
    if (P > 0) then
        ReturnValue ← false
end.{find_n*}
```

Fig. 6. Routine *find_n**.

Step 5: For each backup trip candidate $k \in K$ (corresponding to the assigned backup vehicle candidate), build the corresponding feasible network.

Based on these feasible networks, we can model the SDVRSP as a SDVSP in each feasible network. The optimal schedule of SDVRSP is the one with the minimum total cost over the set of all feasible networks. The overall procedure, which we refer to as SA_VRSP, to obtain the optimal solution to VRSP is based on the combined forward-backward auction algorithm developed by Freling et al. [6].

Algorithm SA_VRSP: Sequential Auction Algorithm for VRSP

Step 1: Based on the starting and ending times of trips and travel time between trips, apply the procedure *Build-Feasible-Networks* to build the set of all possible feasible networks.

Step 2: Calculate the costs for the compatible trip pairs and the total delay cost of each feasible network.

Step 3: For each feasible network in G, apply the forward-backward combined auction algorithm developed in [6] to find the minimum cost scheduling of each feasible network.

Step 4: Select the minimal operating and delay cost scheduling as the solution.

4.4. Implementation

The DSS is designed to run in a platform-independent environment with the Internet browser. The software system uses several programming languages. MySQL v4.0.17 is used to implement the database management module. In order to implement the web-based interface, Apache v1.3.29 is used as the web server. Apache is well known for its efficiency and stability. Moreover, PHP 4.3.4 is chosen as the programming language for the interface. PHP has specially designed tools to access MySQL. A major advantage of our system is that MySQL, PHP and Apache are all free software, and all of them have different versions for multiple platforms. The VSP and VRSP algorithms are implemented using standard C++.

5. Computational experiments

The main objective of the computational experiments is to evaluate the performance of the developed auction-based algorithm, in terms of the required CPU time, to obtain the optimum solution. We used the MIP network solver of the CPLEX, based on the formulation presented in Section 4.3, as the basis for comparison.

The experiments were designed using Carpaneto et al.'s [29] random data generation method for VSP. Let $\rho_1, \rho_2, \ldots, \rho_v$ be *relief points* (i.e., points where trips can start or finish) of a transportation network. We generate them as uniformly random points in a (60×60) square and compute the corresponding travel times $\theta_{a,b}$ as the Euclidean distances between relief points a and b. Concerning the trips, we generate for each trip $T_j (j = 1, \ldots, n)$ the starting and ending relief points, ρ'_j and ρ''_j, as uniformly random integers in $[1, \ldots, v]$. The time between trips T_i and T_j is defined as $\tau_{i,j} = \theta_{\rho''_i \rho'_j}$. The starting and ending times, s_j and e_j, of trip T_j were generated by considering two classes of trips: short trips (with probability 40%) and long trips (with probability 60%). For short trips, b_j is a uniformly random integer in (420,480) with probability 15%, in (480,1020) with probability 70%, and in (1020,1080) with probability 15%, e_j is a uniformly random integer in $(s_j + \tau_{\rho'_j,\rho''_j} + 5, s_j + \tau_{\rho'_j,\rho''_j} + 40)$. For long trips, s_j is a uniformly random integer in (300,1200) and e_j is a uniformly random integer in $(s_j + 180, s_j + 300)$. Costs c_{ij},c_{si} and c_{jt} are defined as follows:

1. $c_{ij} = 10\tau_{i,j} + 2(b_j - e_j - \tau_{i,j})$, for all compatible pairs (T_i, T_j);
2. $c_{si} = 2000$ for depot and trips T_i; and
3. $c_{jt} = \lfloor 10$ (Euclidean distance between depot and $\rho'_j)\rfloor + 2000$ for trip T_j and depot.

In order to compare the computational efficiency of the auction-based algorithm and the MIP solver of CPLEX we consider a situation in which the trips are composed of a combination of short (with probability 40%) and long (with probability 60%) trips. To evaluate the performance of the algorithms, we generate first a VSP problem and solve it. Then, disruption was introduced so that an early trip is chosen as cut trip (trip T_b). Since in real-life situations, determination of backup trips requires knowledge of vehicle capacity and common itineraries whereas in the simulation trips are generated only by distance and travel times, we simply assumed the possible number of backup trips to be among the number {2,3,5,10}. The experiments were carried out on 900 MHz Sun Workstations.

Table 2 compares the performance of both algorithms. The first three columns give the number of remaining trips, the number of backup trips considered, and the average optimal cost. The next two columns show the average CPU seconds, excluding input and output time, for the SA_VRSP and the MIP CPLEX solver. The last column presents the gap between the two solution methods, computed as follows:

$$\mathrm{CPU}_{\mathrm{gap}} = 100 \times \frac{\mathrm{CPU}_{\mathrm{CPLEX}} - \mathrm{CPU}_{\mathrm{SA\_VRSP}}}{\mathrm{CPU}_{\mathrm{CPLEX}}}$$

Larger $\mathrm{CPU}_{\mathrm{gap}}$ implies that the CPU for SA_VRSP is less than the CPU for MIP-CPLEX.

The average CPU time for both solution methods is highly dependent on the problem size. The tables show that for small problems (100 remaining trips) both methods are fast, solving the problem in less than 1s CPU time, even for high number of backup vehicle. The auction-based algorithm is more efficient

for all analyzed situations. On average, algorithm SA_VRSP reduces by 70.12% the required CPU time for CPLEX to find the optimum solution. The auction algorithm is more efficient for small and medium-sized problems (until 1000 remaining trips). As the problem size increases, the auction algorithm slightly reduces its efficiency. In summary, we can conclude that algorithm SA_VRSP is computationally efficient in schedule recovery applications.

6. Case study

To demonstrate the use of the rescheduling DSS, we present a case study based on the solid waste collection services in City of Porto Alegre, Brazil. The solid waste collection involves 150 neighborhoods, with a population of more than 1.3 million. More than 60 tons of solid waste are collected per day and distributed to 8 recycling facilities. The collection and distribution of the solid waste are carried out by a municipal company named DMLU, a Portuguese acronym for Departamento Municipal de Limpeza Urbana (Urban Waste Municipal Department, in English). The recycling facilities are managed by cooperatives, where members are mostly poor and are not part of the mainstream economy. In these facilities, the solid waste is separated, appraised, stored, and commercialized. The profit remains with the cooperatives, making it an important income source for more than 450 workers. As a consequence, the solid waste management program has balanced social and ecological benefits [http://www.lixo.com.br accessed 9 September 2004].

The collection is weekly performed on each street of the city. A requirement is that the solid waste should be on the street for a maximum of 30 min before its collection. Severe fines might be imposed on the residents who do not follow this rule. DMLU distributes informative leaflets giving the schedule of the collection trucks for each street. The main purpose is to protect the profits of the cooperatives that run the recycling facilities, since other independent companies may collect the waste before DMLU.

A team of solid waste collectors is composed of one driver and three garbage collectors. There are 24 specially designed trucks to support the waste collection. On average, 23 trucks are used during each shift.

Although the system is efficient in terms of collection, the costs involved are high (around 35% of the DMLU budget). Uncertain situations, such as vehicle breakdowns and problems in the recycling facilities, have not been effectively incorporated. Consequently, managers in DMLU are concerned about the efficiency of the existing waste-flow methods. Furthermore, the current system generates unacceptable schedules in terms of imbalanced trip assignments to recycling facilities (where some facilities may be allocated excessive collection trips, while other facilities may be idle), which are rejected by the human scheduler.

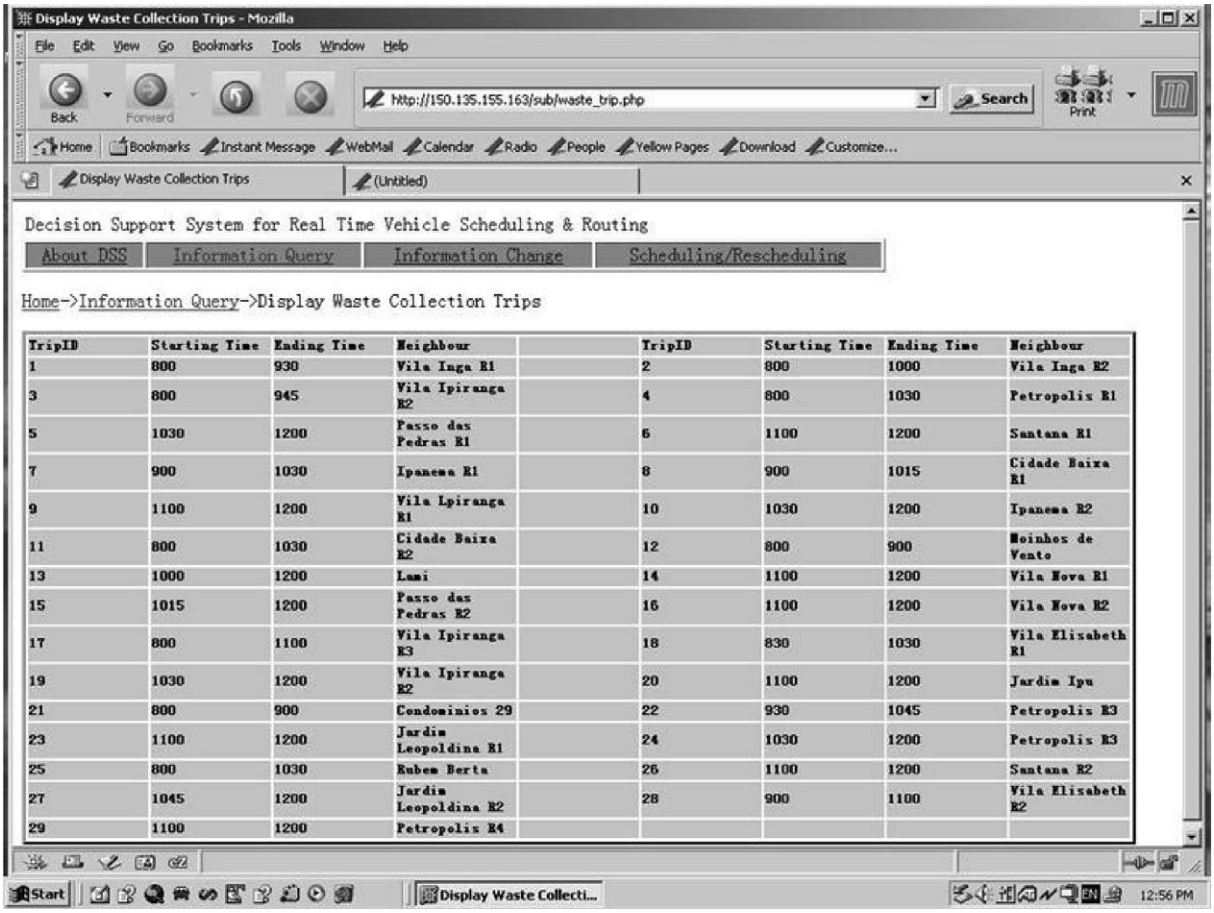
6.1. Experiment configuration

The goal of the case study was to demonstrate the potential of the DSS as an effective and efficient operational planning tool for this logistics problem. The specific objective of the study was to answer the following three questions: (1) What is the best routing and scheduling plan using the dedicated collection vehicles? (2) What is a good routing pattern that has both good operating costs and balanced trip assignment to each recycling facility? (3) What is the best scheduling strategy when an unexpected event occurs and the original plan cannot be used?

Table 2
Simulation results

Remaining trips	Backup trips	Objective function value	Average CPU Time (s)		CPU_{gap}(%)
			SA_VRSP	CPLEX	
100	2	137244	0.038	0.162	76.54
	3	137182	0.055	0.244	77.46
	5	137138	0.091	0.409	77.75
	10	135804	0.180	0.808	77.72
300	2	344513	0.394	1.819	78.34
	3	344477	0.607	2.726	77.73
	5	341700	1.022	4.549	77.53
	10	341700	2.078	9.133	77.25
500	2	558427	1.507	5.976	74.78
	3	558422	2.258	8.973	74.84
	5	558372	3.752	14.952	74.91
	10	558326	7.623	29.914	74.52
700	2	752984	3.621	12.573	71.20
	3	752882	5.399	18.900	71.43
	5	752863	9.051	31.493	71.26
	10	752824	18.126	63.040	71.25
900	2	950713	6.779	23.564	71.23
	3	950695	10.157	35.394	71.30
	5	950625	17.002	58.933	71.15
	10	950281	35.629	117.794	69.75
1100	2	1140430	16.098	39.081	58.81
	3	1140390	23.971	58.557	59.06
	5	1139960	40.116	97.520	58.86
	10	1139940	80.125	194.906	58.89
1300	2	1346500	23.157	59.004	60.75
	3	1346470	35.162	88.568	60.30
	5	1346400	59.395	147.449	59.72
	10	1346370	120.564	295.093	59.14

In order to study these questions, we first use the time schedules of all trips and travel time of deadheading trips to construct a feasible network for solving VSP; this answers the first question. Then, the number of trips which is assigned to each recycling facility is checked. If the assignment is not balanced, the capacity issue is considered for each facility. A variation of auction algorithm is called to solve the problem. Finally, to answer the third question, disruption was simulated where an early trip is chosen



TripID	Starting Time	Ending Time	Neighbour		TripID	Starting Time	Ending Time	Neighbour
1	800	930	Vila Inga R1		2	800	1000	Vila Inga R2
3	800	945	Vila Ipiranga R2		4	800	1030	Petropolis R1
5	1030	1200	Passo das Pedras R1		6	1100	1200	Santana R1
7	900	1030	Ipanema R1		8	900	1015	Cidade Baixa R1
9	1100	1200	Vila Ipiranga R1		10	1030	1200	Ipanema R2
11	800	1030	Cidade Baixa R2		12	800	900	Moinhos de Vento
13	1000	1200	Lami		14	1100	1200	Vila Nova R1
15	1015	1200	Passo das Pedras R2		16	1100	1200	Vila Nova R2
17	800	1100	Vila Ipiranga R3		18	830	1030	Vila Elisabeth R1
19	1030	1200	Vila Ipiranga R2		20	1100	1200	Jardim Ipu
21	800	900	Condominios 29		22	930	1045	Petropolis R3
23	1100	1200	Jardim Leopoldina R1		24	1030	1200	Petropolis R3
25	800	1030	Ruben Berta		26	1100	1200	Santana R2
27	1045	1200	Jardim Leopoldina R2		28	900	1100	Vila Elisabeth R2
29	1100	1200	Petropolis R4					

Fig. 7. Collection trips conducted on Monday morning.

as cut trip. Then candidates of backup vehicles are selected and corresponding feasible networks are generated. The developed algorithm for VRSP is called to solve this problem.

The collection data for Monday morning was selected for our case study (see Fig. 7). All computational experiments were conducted on a 900 MHz Sun Workstation.

6.2. Modeling the problem

The solid waste collection problem can be treated as a VSP/VRSP problem with a special structure. Initially, all vehicles leave the depot to serve the collection trips. After serving a collection trip, the vehicle must unload the collected solid waste in a recycling facility. Then, the vehicle can go back to the depot or continue to serve a new compatible trip, if it can arrive in the starting place of the next trip before its starting time. Two situations are not allowed in this service: (1) a vehicle comes from the depot and directly goes to the recycling facility, and (2) after finishing its collection trip, a vehicle directly goes to the depot or goes to serve a new trip without unloading its cargo at a recycling facility.

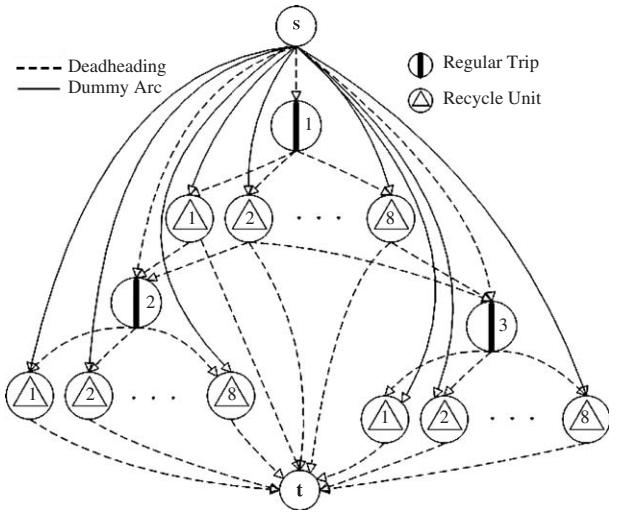


Fig. 8. An example of feasible network structure for the solid waste collection problem.

First, we need to construct a feasible network structure that simultaneously represents the problem and is in accordance with the algorithms implemented within the DSS. In this structure both the collection trips and recycling facilities are represented as nodes. For each collection trip, we created eight associated dummy trips which represent eight recycling facilities respectively. Fig. 8 illustrates the network structure for this problem. The starting time of each *recycling facility associated* (RFA) trip of a is the ending time of the corresponding collection trip plus the travel time from the ending point of its collection trip to the recycling facility. The duration of each RFA trip is the unloading and service times at the facility. After determining the starting and ending times of each such RFA trip, we can construct the feasible network in such a way that a collection trip is connected to all recycling facilities through eight RFA trips. There are no direct connections among collection trips. Since in vehicle scheduling problems, every trip needs to be covered, we introduce dummy arcs connecting the depot to all recycling facility nodes, and from these nodes to the depot (see Fig. 8). As a consequence, several dummy vehicle assignments are carried out only to fulfill this VSP peculiarity. These dummy assignments are easily discarded in the final solutions. For example, in Fig. 8, trip 1 has eight RFA trips. Vehicles from the depot can go to trip 1, therefore there is an arc from depot to trip 1. After serving trip 1, vehicle has to go to recycling facility to unload the waste. Therefore, there are arcs from trip 1 to eight RFA trips. Since it is not allowed for a vehicle to go back to depot directly unless the waste is unloaded at the recycling facility, there is no arc from trip 1 to depot. After unloading the waste at a recycling facility, the vehicle can go back to the depot or serve next compatible trips. Thus, there is an arc from the recycling facility to the depot, and there are also arcs from recycling facility to trips 2 and 3. Since every recycling facility is represented by a RFA trip, there are arcs from the depot to all RFA trips so that all of them can be covered.

It should be noted that in order to simplify the analysis, we consider the distance traveled by each collection truck as the only variable operational cost involved. All remaining cost items are defined as fixed costs. This is coherent with the public service characteristic of DMLU, where the costs related with personnel

Table 3
Solutions for the VSP

Vehicle	Solution 1	Solution 2	Solution 3	Solution 4	Solution 5
V1	T1-R8-T15-R5	T1-R8-T5-R8	T1-R8-T24-R8	T1-R8-T24-R8	T1-R4-T13-R6
V2	T2-R8-T24-R8	T2-R8-T24-R8	T2-R8-T27-R8	T2-R6-T19-R4	T2-R8-T24-R8
V3	T3-R1-T5-R8	T3-R6-T19-R2	T3-R6-T19-R6	T3-R6-T13-R6	T3-R6-T19-R6
V4	T4-R1-T9-R1	T4-R2-T29-R7	T4-R1-T9-R4	T4-R3-T16-R1	T4-R7-T27-R8
V5	T6-R6	T6-R7	T5-R8	T5-R8	T5-R8
V6	T7-R5-T14-R5	T7-R5-T14-R5	T6-R3	T6-R7	T6-R7
V7	T8-R1-T19-R1	T8-R3-T27-R8	T7-R2-T14-R5	T7-R5-T14-R5	T7-R5-T14-R5
V8	T10-R5	T10-R2	T8-R3	T8-R3-T9-R1	T8-R1
V9	T11-R1-T23-R1	T11-R3-T9-R6	T10-R2	T10-R2	T10-R4
V10	T12-R1-T13-R6	T12-R6-T13-R6	T11-R3-T23-R4	T11-R4-T23-R7	T11-R3-T9-R1
V11	T17-R5	T15-R5	T12-R6-T13-R4	T12-R6-T22-R3	T12-R6-T22-R1-T20-R8
V12	T18-R8-T16-R8	T17-R5	T15-R5	T15-R5	T15-R5
V13	T21-R1-T22-R1-T20-R8	T18-R1-T16-R1	T17-R5	T17-R2	T17-R4
V14	T25-R8-T27-R8	T21-R7-T22-R1-T20-R8	T18-R1-T16-R4	T18-R1-T20-R7	T18-R1-T16-R1
V15	T26-R1	T25-R1-T23-R4	T21-R4-T22-R4	T21-R4	T21-R7
V16	T28-R8	T26-R4	T25-R6-T20-R1	T25-R8-T27-R8	T25-R7-T23-R7
V17	T29-R1	T28-R1	T26-R4	T26-R5	T26-R7
V18			T28-R6	T28-R1	T28-R1
V19			T29-R4	T29-R7	T29-R7
Distance (km)	455.00	457.10	452.90	462.50	488.30
Total Costs	295.79	296.05	323.52	324.74	328.01

and infra-structure do not depend on the productivity of the system. The total cost is defined as follows:

$$C = \sum_{i=1}^{V} c_i d_i + \sum_{i=1}^{V} f_i,$$

where c_i is the average operational cost of vehicle based on its diesel fuel consumption (we used a value of US$ 0.127/Km, considering the current diesel price in Brazil and the average diesel consumption of the vehicles), f_i is the average diary fixed cost (this includes taxes, maintenance, etc.) of vehicle i (defined as a constant equal to US$ 14.00, based on DMLU databases), d_i is the distance travelled by vehicle i (excluding waste collection trips), and V is the total number of vehicles used to collect the solid waste.

6.3. Solving vehicle scheduling

Tables 3 and 4 present the results for the scheduling decisions. Table 3 gives a summary of the resulting schedules, while Table 4 presents the assignment of collection trips to each recycling facility. In all tables, T denotes a collecting trip, V_i denotes a vehicle i, and R_j denotes a recycling facility j. Solution 1 shows the initial optimal schedule defined by the DSS. Although this solution obtains the minimum cost, it cannot be employed since several recycling facilities are not used, and the optimal solution is imbalanced in terms of the use of the recycling facility capacities (see Table 4). Recycling facilities R2, R3, R4 and R7 are not used in this schedule, while R1 receives 9.2 tons, comparing with the average 7.5 tons collected

Table 4
Trip distributions in each recycling facility

Facility	Solution 1	Solution 2	Solution 3	Solution 4	Solution 5
1	T3 T4 T8 T9 T11 T12 T19 T21 T22 T23 T26 T29	T16 T18 T22 T25 T28	T4 T18 T20	T9 T16 T18 T28	T8 T9 T16 T18 T22 T28
2		T4 T10 T19	T7 T10	T10 T17	
3		T8 T11	T6 T8 T11	T4 T8 T22	T11
4		T23 T26	T9 T13 T16 T21 T22 T23 T26 T29	T11 T19 T21	T1 T10 T17
5	T7 T10 T14 T15 T17	T7 T14 T15 T17	T14 T15 T17	T7 T14 T15 T26	T7 T14 T15
6	T6 T13	T3 T9 T12 T13	T3 T12 T19 T25 T28	T2 T3 T12 T13	T3 T12 T13 T19
7		T6 T21 T29		T6 T20 T23 T29	T4 T6 T21 T23 T25 T26 T29
8	T1 T2 T5 T16 T18 T20 T24 T25 T27 T28	T1 T2 T5 T20 T24 T27	T1 T2 T5 T24 T27	T1 T5 T24 T25 T27	T2 T5 T20 T24 T27

```
routine DynamicPenalty:
begin
    for each recycling facility r
    begin
        Determine the numbers of trips assigned to it
        if the numbers of assigned trips exceed the given limit, then
            Add a penalty cost to the unassigned arcs which are connected to r
    end
end.{DynamicPenalty}
```

Fig. 9. Routine *Dynamic Penalty*.

by each trip (based on DMLU databases). This imbalanced scheduling solution is unacceptable by the human schedulers taking into consideration the social benefits of the solid waste program.

In order to obtain a more suitable solution, which compromises the social and financial aspects involved, we decided to restrict the number of trips assigned to each recycling facility. Penalties were introduced in the auction algorithm implemented in the DSS to solve the VSP. Each time the number of trips serving a recycling facility exceeds a given limit, all remaining arcs connecting trips to this specific recycling facility are penalized, making them less attractive in the final solution. Specially, the following routine *DynamicPenalty* (see Fig. 9) was included of the auction-based algorithm developed by Freling et al. [6] at the beginning of each iteration. Here penalties are dynamically imposed on arcs as the assignments are performed at any iteration of a sub-problem. We decided to use penalties instead of constraints in order to

avoid infeasibility. The given limit of each recycling facility is related to its capacity and the total number of collection trips. Solution 2 in Table 2 presents the solution obtained using penalties with 4 as the given limit for each recycling facility. Although the number of trips assigned to each recycling facility is not equal, there is a significant improvement in the results. All recycling facilities are now used (see Table 4).

We attempted to improve the solution by cutting arcs between trips and recycling facilities, and increasing/decreasing the number of trips using the less/over utilized facilities. Solutions 3–5 are the results obtained for some cuts introduced in Solution 2. Solution 3 corresponds to a network in which all trips, except T1, T5, T2, T24 and T27, are not connected to recycling facility 8. Solution 4 corresponds to the network in which trips T20 and T25 are not connected to recycling facilities 8 and 1, respectively; while solution 5 corresponds to situation in which the number of trips allowed to use recycling facilities 3 and 4 are increased to 6. None of the several attempts offered better results than solution 2 in terms of total costs and balanced use of the recycling facility capacities. The number of vehicles increases from 17 to 19 in solutions 3, 4 and 5. Although solutions 4 and 5 decreases the total distance traveled by the vehicles, they are not interesting due to the need of extra vehicles, which increases the total costs. Solutions 3–5 were therefore rejected by the human schedulers. Solution 2 was considered as the most suitable schedule.

A human scheduler also developed a schedule manually. Solution 2 reduces significantly the costs components in comparison with the solution of human scheduler. The number of vehicles reduced from 23 to 17 vehicles; and the traveled distance excluding waste collection trips decreased from 649.70 to 457.10 km; as a result, the total cost was reduced from US$ 404.51 to US$ 296.05, resulting in an estimated saving of 26.81% for Monday mornings. Furthermore, the manual solution used only five recycling facilities, leaving three units not being utilized.

6.4. Solving vehicle rescheduling

The following situation was addressed to answer what it is the best plan of action when an unexpected event occurs. A possible breakdown of vehicle serving trip T3 at 8:50 was introduced in the initial schedule proposed by the VSP (solution 2 in Table 3). The breakdown data was based on previous statistics maintained by the DMLU. It should be noted that we assume that a vehicle in a collection trip can change its recycling facility destination, but vehicles in deadheading trips should deliver the collected waste at the recycling facility defined in solution 2.

Since only a vehicle covers a specific street, itinerary compatible trips do not exist in this problem. Therefore, in the algorithm *Build-Feasible-Networks*, set A is empty, and step 2 does not have a solution. In this situation, B^* is treated as B. The human schedulers decided to set $T_l = 9{:}15$, since a higher value creates serious disturbances in the collection process. In this case, the backup vehicles candidates comprise vehicles serving trip T12, trip T21, and an extra vehicle from depot.

Algorithm SA_VRSP, including routine *DynamicPenalty* in step 3 (to obtain a better balance in the number of trips assigned to each recycling facility), was used to solve this disruption decision making process. Tables 5 and 6 present the solutions obtained for the following three alternatives for the backup vehicle: (i) the vehicle serving trip T12; (ii) the vehicle serving trip T21; and (iii) an extra vehicle from depot. The total cost for each backup trip candidate k is

$$C_k = \sum_{i=1}^{V} c_i d_i + \sum_{i=1}^{V} f_i + cD_k,$$

Table 5
New schedules considering a breakdown in vehicle serving trip T3

Vehicle	Backup trip T12	Backup trip T21	Backup vehicle from depot
V1	T1-R8-T24-R8	T1-R6-T13-R6	T1-R6-T24-R8
V2	T2-R5-T14-R5	T2-R6-T19-R6	T2-R4-T19-R6
V3	T5-R1	T5-R8	T3-R4-T5-R8
V4	T4-R4-T6-R3	T4-R4-T23-R4	T4-R1-T20-R4
V5	T15-R5	T6-R4	T6-R4
V6	T7-R2-T23-R4	T7-R5-T14-R5	T7-R5-T14-R5
V7	T8-R3-T9-R1	T8-R3-T27-R8	T8-R3-T27-R8
V8	T10-R2	T10-R2	T10-R4
V9	T11-R3-T20-R3	T11-R4-T29-R3	T11-R3-T23-R5
V10	T12-R6-T3-R6-T13-R3	T12-R4-T22-R7-T20-R7	T12-R4-T22-R3
V11	T19-R6	T15-R5	T15-R5
V12	T17-R2	T17-R2	T17-R4
V13	T18-R8-T27-R8	T18-R7-T16-R6	T18-R1-T16-R1
V14	T21-R6-T22-R2	T21-R7-T3-R7-T24-R8	T21-R7-T13-R6
V15	T25-R1-T16-R8	T25-R1-T9-R1	T25-R1-T9-R6
V16	T26-R3	T26-R7	T26-R3
V17	T28-R6	T28-R6	T28-R1
V18	T29-R4		T29-R4
Distance (km)	446.70	453.10	496.80
Delay (min)	1	3	10
Total costs	308.98	296.29	317.59

Table 6
Trip distributions in each recycling facility

Facility	Backup trip T12	Backup trip T21	Backup vehicle from depot
1	T5 T9 T25	T9 T25	T4 T16 T18 T25 T28
2	T7 T10 T17 T22	T10 T17	
3	T6 T8 T11 T13 T20 T26	T8 T29	T8 T11 T22 T26
4	T4 T23 T29	T4 T6 T11 T12 T23	T2 T3 T6 T10 T12 T17 T20 T29
5	T2 T14 T15	T7 T14 T15	T7 T14 T15 T23
6	T3 T12 T19 T21 T28	T1 T2 T13 T16 T19 T28	T1 T9 T13 T19
7		T3 T18 T20 T21 T22 T26	T21
8	T1 T16 T18 T24 T27	T5 T24 T27	T5 T24 T27

where c is the delay cost per unit time (the human schedulers defined it as US$ 0.25/min, considering the problem peculiarities), and D_k is the total delay (in min) for backup trip k.

The best solution is with the backup vehicle serving trip T21. This solution gives the lowest total cost, an acceptable delay time (3 min) in accordance with the human schedulers, uses the same number of vehicles as solution 2 (17 vehicles), and has the best distribution of waste among the recycling facilities (see Table 6). Nevertheless, R4, R6, and R7 are overloaded in comparison to R1, R2, R3, and R8. We compared the solutions given by human schedulers for this specific situation. They usually send a new

vehicle from depot to backup the cut trip. In general, all trips in which this vehicle is scheduled are delayed (at least for an hour) or cancelled, causing severe effects in the DMLU's level of service. When this happens, DMLU receives several phone calls from residents and the recycling facility managers, complaining about the situation. Unfortunately, it is impossible to compute the financial impacts, since DMLU does not record data on disruptions.

Observe that the three scenarios introduced several changes in the initial schedule. Considering vehicle schedules, the new schedule with trip T12 as the backup trip changes all vehicle schedules except the vehicles serving trips T10 and T15. For the solution with backup vehicle from depot and the solution with backup trip T21, only four vehicles keep their initial schedules. Observe also, as a consequence of vehicle schedule changes, the trip distributions for each recycling facility also change. These changes can make the crew rescheduling problem difficult, since we have to guarantee that all teams know the itinerary of each collection trip and how to drive to recycling facilities after the trips. This is not so serious in this case study, since all teams know all trip itineraries. However, it may not always be the case. Although it is almost impossible to obtain a solution for the VRSP without effecting the initial schedule, it is possible, in order to decrease the number of possible changes in the initial schedule, to introduce penalties in the cost of some arcs of the feasible networks. The implementation of these penalties will require modifications in the auction algorithm. This is a topic of future research, with the objective of increasing the modeling capabilities of the DSS.

In terms of computation times, we did not perform precise measurements on the test problems. Based on approximate estimates, the VSP and the VRSP algorithms require 3–20 CPU seconds and 5–25 CPU seconds, respectively, depending on the size of problems being solved. These times can be considered acceptable given the complexity of the problems, and the human schedulers were very satisfied with the DSS solution times.

Overall, the human schedulers responded positively to the prototype. The main advantage of the DSS was to offer a tool for objective analysis, avoiding the evaluation of scheduling/rescheduling options being based only on empirical factors or only simple operational and financial measures. Analysis and evaluation of the possible scheduling/rescheduling alternatives, through our model-based DSS, provides a means to study each alternative with respect to several measures and make more objective decisions. The possibility to alter operational scenarios interactively, to evaluate them in an efficient and effective manner, and to include real-world constraints and limitations make the DSS an effective tool for schedule recovery.

7. Conclusions

This paper describes a decision support system for the single-depot vehicle scheduling/rescheduling problem. The main objective of the DSS is to help human schedulers to find optimal schedules with and without unexpected incidents. The DSS can play an important role in the operational planning of transportation/logistics companies.

The developed DSS introduces a systematic procedure for prescriptive decision-making for both scheduling and rescheduling due to disruptions. The approach helps to solve the complex problem of recovery from severely disrupted trips. Computational experiments to evaluate the performance of the algorithms, by comparing the developed algorithm with the MIP solver of CPLEX, using randomly generated data, show that the DSS has potential as an effective and efficient tool for real-time operational

planning in transportation/logistic companies. A case study was used to illustrate the developed DSS and its use in an actual system.

Future research is directed towards the following: (i) the expansion of the modeling capabilities with the inclusion of additional constraints in our VRSP formulation, restricting the number of trips that are rescheduled for a vehicle; (ii) the integration of the crew scheduling problem into the DSS; and (iii) the improvement of the computational capabilities of the DSS.

Acknowledgements

This paper was written while the second author visited the ATLAS Center at the University of Arizona, through funding by CAPES, Brazil. The first and third authors acknowledge the support received for this research from USDOT/FHWA and Arizona DOT through their sponsorship of the ATLAS Programs.

References

[1] Meecham M. Airport traffic increase reflects airline recovery. Aviation Week and Space Technology 1995;143(5):36.
[2] Ichoua S, Gendreau M, Potvin JY. Diversion issues in real-time vehicle dispatching. Transportation Science 2000;34: 426–38.
[3] Baita F, Pesenti R, Ukovich W, Favaretto D. A comparison of different solution approaches to the vehicle scheduling problem in a practical case. Computers and Operations Research 2000;27:1249–69.
[4] Bodin L, Golden B. Classification in vehicle routing and scheduling. Networks 1981;11:97–108.
[5] Daduna JR, Paixão JM. Vehicle scheduling for public mass transit—an overview. In: Proceedings of sixth international conference on computer-aided scheduling of public transport. Boston, MA; 1995. p. 76–90.
[6] Freling R, Wagelmans A, Paixão J. Models and algorithms for single-depot vehicle scheduling. Transportation Science 2001;35:165–80.
[7] Bokinge U, Hasselstrom D. Improved vehicle scheduling in public transport through systematic changes in the time-table. European Journal of Operational Research 1980;5:388–95.
[8] Amico D, Fischetti M, Toth P. Heuristic algorithms for the multiple depot vehicle scheduling problem. Management Science 1993;39:115–25.
[9] Jonker R, Volgenant T. Improving the Hungarian assignment problem. Operations Research Letters 1986;5:171–6.
[10] Song T, Zhou L. A new algorithm for the quasi-assignment problem. Annals of Operations Research 1990;37:205–23.
[11] Paixão JM, Branco I. A quasi-assignment algorithm for bus scheduling. Networks 1987;17:249–69.
[12] Bertsekas D, Eckestein J. Dual coordinate step methods for linear network flow problems. Mathematical Programming 1988;42:203–43.
[13] Psaraftis HN. Dynamic vehicle routing: status and prospects. Annals of Operations Research 1995;61:143–64.
[14] Ghiani G, Guerriero F, Laporte G, Musmanno R. Real-time vehicle routing: solution concepts, algorithms and parallel computing strategies. European Journal of Operational Research 2003;151(1):1–11.
[15] Gendreau M, Potvin JY. Dynamic vehicle routing and dispatching. In: Crainic T, Laporte G, editors. Fleet Management and Logistics. New York: Kluwer; 1998. p. 115–26.
[16] Yang J, Jaillet P, Mahmassani H. Real-time multivehicle truckload pickup and delivery problems. Transportation Science 2004;38:135–48.
[17] Laporte G, Louveaux FV. Solving stochastic routing problems with integer L-shaped method. In: Crainic T, Laporte G, editors. Fleet management and logistics. New York: Kluwer; 1998. p. 159–67.
[18] Powell WB, Jaillet, P, Odoni A. Stochastic and dynamic networks and routing. In: Ball MO, Magnanti TL, Monma CL, Nemhauser GL, editors. Handbooks in operations research and management science, Network Routing. Amsterdam, The Netherlands: North-Holland; 1995. p. 141–295.
[19] Carlson PM. Exploiting the opportunities of collaborative decision making: a model and efficient solution algorithm for airline use. Transportation Science 2000;34:381–93.

[20] Lettovský L. Airline operations recovery: an optimization approach. PhD thesis. Georgia Institute of Technology USA, 1997.
[21] Rosenberger JM, Johnson EL, Nemhauser GL. Rerouting aircraft for airline recovery. Transportation Science 2003;37: 408–21.
[22] Teodorović D, Stojković G. Model to reduce airline schedule disturbances. Journal of Transportation Engineering 1995;121:324–31.
[23] Huisman D, Freling R, Wagelmans A. A robust solution approach to the dynamic vehicle scheduling problem. Transportation Science 2004;38:447–58.
[24] Li JQ, Mirchandani PB, Borenstein D. Parallel auction algorithm for bus rescheduling. In: Proceedings of ninth international conference on computer-aided scheduling of public transport. California, USA: San Diego; 2004.
[25] Desrochers M, Lenstra JK, Savelsbergh MWP, Stougie L. Towards a model and algorithm management system for vehicle routing and scheduling problems. Decision Support Systems 1999;25:109–13.
[26] Basnet C, Foulds L, Igbaria M. Fleet Manager: a microcomputer-based decision support system for vehicle routing. Decision Support Systems 1996;16:195–207.
[27] Nussbaum M, Sevulpeda M, Cobian A, Gaete J, Parra E, Cruz J. A fuel distribution knowledge-based decision support system. Omega 1997;25:225–34.
[28] Ruiz R, Maroto C, Alcaraz J. A decision support system for a real vehicle routing problem. European Journal of Operational Research 2004;153:593–606.
[29] Carpaneto G, Dell'Amico M, Fischetti M, Toth P. A branch and bound algorithm for the multiple depot vehicle scheduling problem. Networks 1989;19:531–48.