UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

ESCOLA DE ADMINISTRAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM ADMINISTRAÇÃO
DOUTORADO EM ADMINISTRAÇÃO

Essays on Urban Bus Transport Optimization

Pablo Cristini Guedes

Porto Alegre, June 27, 2017.

Pablo Cristini Guedes

Essays on Urban Bus Transport Optimization

Dissertation for Doctoral degree in Business Administration at the School of Administration of Federal University of Rio Grande do Sul.

Supervisor: Denis Borenstein, PhD

Brazil

June 27, 2017

CIP - Catalogação na Publicação

Guedes, Pablo
   Essays on Urban Bus Transport Optimization / Pablo Guedes. -- 2017.
   86 f.

   Orientador: Denis Borenstein.

   Tese (Doutorado) -- Universidade Federal do Rio Grande do Sul, Escola de Administração, Programa de Pós-Graduação em Administração, Porto Alegre, BR-RS, 2017.

   1. Vehicle Scheduling. 2. Vehicle Rescheduling. 3. Urban Bus Transportation. I. Borenstein, Denis, orient. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da UFRGS com os dados fornecidos pelo(a) autor(a).

Pablo Cristini Guedes

Essays on Urban Bus Transport Optimization

Dissertation for Doctoral degree in Business Administration at the School of Administration of Federal University of Rio Grande do Sul.

Denis Borenstein, PhD
Dissertation Advisor

Bruno de Athayde Prata, PhD
Committee Member

Tiago Pascoal Filomena, PhD
Committee Member

Michel José Anzanello, PhD
Committee Member

Brazil

June 27, 2017

"É preciso ter um caos dentro de si para dar à luz uma estrela cintilante."

— Nietzsche

Agradecimentos

Ainda não "me caiu a ficha" que estou finalizando essa etapa da minha vida. Foram aproximadamente 5 anos e meio de mestrado e doutorado. Hoje mais cedo enquanto nervoso estava para a minha defesa, minha amada esposa me perguntou por que eu tinha escolhido fazer mestrado e doutorado? Eu sem pestanejar respondi: "Porque eu quero ser professor". Essa profissão que nós não escolhemos, é ela que nos escolhe. Hoje após uma longa jornada estou finalizando mais uma etapa da minha vida e tanto difícil quanto é a caminhada, foi grande a ajuda que recebi de vocês e por isso gostaria de agradecer a todos.

Portanto agradeço, primeiramente, à minha amada esposa Marina Delanni Vitória Guedes, pelos diversos dias difíceis que passaste me dando apoio e carinho, sempre com muita compreensão e dedicação. Obrigado pelo auxílio, compreendendo a importância dessa etapa nas nossas vidas e dando o suporte necessário.

Agradeço à minha família, especialmente aos meus pais e minha irmã, que sempre me apoiaram e me deram suporte para conseguir chegar até aqui. Me deram valores, educação (essa talvez com muito sacrifício) e me possibilitaram chegar nessa conquista. Ela é parte de vocês. Saiba que eu sempre serei grato a vocês.

Quero um especial agradecimento ao meu orientador, padrinho de casamento e amigo Professor Denis. Foste um orientador completo, me auxiliando no trabalho, mas me ajudando a evoluir como ser humano. Foste um educador que não se limitou a me auxiliar como orientador, mas me transformou como profissional. Muito obrigado!

Gostaria de agradecer a todos os meu amigos dessa jornada que consquistei. Não irei nomeá-lo para não cometer injustiças. Me limito a dizer que foi essencial na minha formação as constantes discussões, conversas e aprendizados que compartilhei com vocês da 329.

Agradeço aos colegas de trabalho e artigos que compartilharam uma enorme parte deste trabalho, sempre me auxiliando e ajudando a dismistificar os artigos.

Um agradecimento a todos os professores do PPGA de Pesquisa Operacional pelos ensinamentos, em especial aos membros da banca Prof. Bruno Prata, Prof. Tiago Filomena, Michel Anzanelllo pelas contribuições, comentários, elogios e por aceitar participar dessa etapa comigo.

A Capes pela ajuda financeira, pois sem a bolsa este trabalho não chegaria a este nível de excelência.

Enfim agradeço a todos que de alguma forma contribuiram para esse trabalho. Um Abraço.

Abstract

In this dissetation we presented a three articles compilation in urban bus transportation optimization. The main objective was to study and implement heuristic solutions method based on Operations Research to optimizing offline and online vehicle (re)scheduling problems considering multiple depots and heterogeneous fleet. In the first paper, a fast heuristic approach to deal with the multiple depot vehicle scheduling problem was proposed. We think the main contributions are the column generation framework for large instances and the state-space reduction techniques for accelerating the solutions. In the second paper, we added complexity when considering the heterogeneous fleet, denoted as "the multiple-depot vehicle-type scheduling problem" (MDVTSP). Although the MDVTSP importance and applicability, mathematical formulations and solution methods for it are still relatively unexplored. We think the main contribution is the column generation framework for instances with heterogeneous fleet since no other proposal in the literature has been identified at moment by the authors. In the third part of this dissertation, however, we focused on the real-time schedule recovery for the case of serious vehicle failures. Such vehicle breakdowns require that the remaining passengers from the disabled vehicle, and those expected to become part of the trip, to be picked up. In addition, since the disabled vehicle may have future trips assigned to it, the given schedule may be deteriorated to the extent where the fleet plan may need to be adjusted in real-time depending on the current state of what is certainly a dynamic system. Usually, without the help of a rescheduling algorithm, the dispatcher either cancels the trips that are initially scheduled to be implemented by the disabled vehicle (when there are upcoming future trips planned that could soon serve the expected demand for the canceled trips), or simply dispatches an available vehicle from a depot. In both cases, there may be considerable delays introduced. This manual approach may result in a poor solution. The implementation of new technologies (e.g., automatic vehicle locators, the global positioning system, geographical information systems, and wireless communication) in public transit systems makes it possible to implement real-time vehicle rescheduling algorithms at low cost. The main contribution is the efficient approach to rescheduling under a disruption. The approach with integrated state-space reduction, initial solution, and column generation framework enable a really real-time action. In less than five minutes rescheduling all trips remaining.

Keywords: Multiple Depot Vehicle Scheduling, Heterogeneous Fleet, Rescheduling, Public Transport.

Resumo

Nesta tese, nós apresentamos uma compilação de três artigos de otimização aplicados no contexto de transporte urbano de ônibus. O principal objetivo foi estudar e implementar heurísticas com base em Pesquisa Operacional para otimizar problemas de (re)escalonamento de veículos off-line e on-line considerando várias garagens e frota heterogênea. No primeiro artigo, foi proposta uma abordagem heurística para o problema de escalonamento de veículos múltiplas garagens. Acreditamos que as principais contribuições são o método de geração de colunas para grandes instâncias e as técnicas de redução do espaço de estados para acelerar as soluções. No segundo artigo, adicionamos complexidade ao considerar a frota heterogênea, denotada como *multiple depot vehicle type scheduling problem* (MDVTSP). Embora a importância e a aplicabilidade do MDVTSP, formulações matemáticas e métodos de solução para isso ainda sejam relativamente inexplorados. A principal contribuição desse trabalho foi o método de geração de colunas para o problema com frota heterogênea, já que nenhuma outra proposta na literatura foi identificada no momento pelos autores. Na terceira parte desta tese, no entanto, nos concentramos no reescalonamento em tempo real para o caso de quebras definitivas de veículos. A principal contribuição é a abordagem eficiente do reescalonamento sob uma quebra. A abordagem com redução de espaço de estados, solução inicial e método de geração de colunas possibilitou uma ação realmente em tempo real. Em menos de cinco minutos, reescalonando todas as viagens restantes.

Palavras-chave: Escalonamento de veículos com múltiplas garagens, frota heterogênea, rescalonamento, transporte urbano.

Contents

Introduction

Literature refers to planning processes involving all decisions that precede an operation as Transit Network Planning (TNP). Due to its complexity, TNP is often divided into sub-steps (or subproblems) encompassing strategic decisions, operational decisions, and all classes of decision in between: (i) Transit Network Design (TND); (ii) Transit Network Timetabling (TNT); (iii) Vehicle Scheduling (VS); (iv) Crew Scheduling (DS); (v) Driver Rostering (DR) and Real-time Control (RT). Figure 1 shows interrelations between problems in TNP.

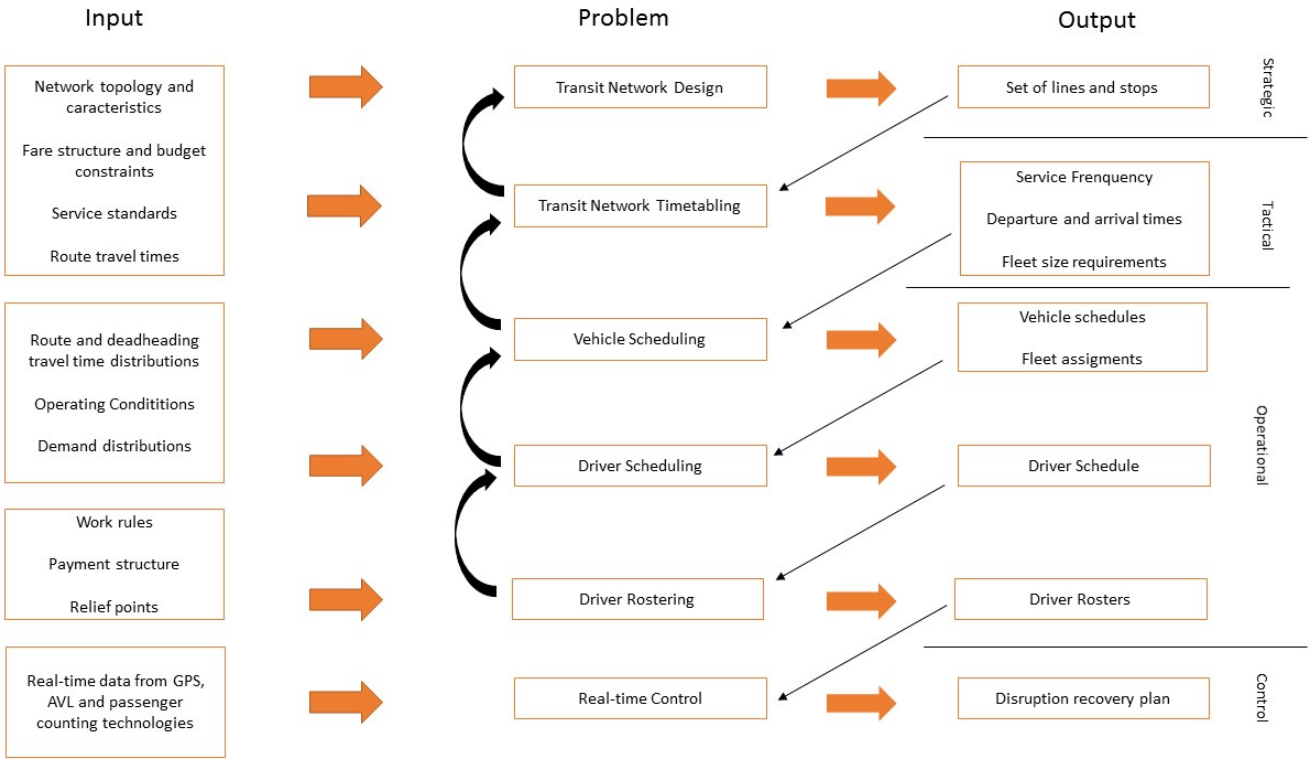

Figure 1 – Interaction between stages of the planning process.

The current dissertation focuses vehicle scheduling and rescheduling. This research developed an online and offline scheduling. More specifically, this project aims to address urban bus transportation problems through three subprojects that stand on their own, but also complement each other to form a consistent, meaningful body of contributions to the overarching urban transportation theme. These subprojects will be presented as individual articles.

Efficiency in urban mobility is no longer a desirable feature in large cities, but a true necessity. Uncontrolled urban sprawl leads to a series of transportation-related problems (e.g., congestion in metropolitan and even in suburban areas, pollution, increases

the number of accidents, disruptions like delays, vehicle breaks). Developing more efficient, high-quality public transportation systems might be a sustainable, affordable alternative to this situation. One approach to enhancing our transportation systems builds on the optimization paradigm. However, optimization problems in public transportation of large cities have proven to be of extreme complexity due to the number of daily trips and the disruption possibilities (VISENTINI et al., 2014). For instance, according to Ibarra-Rojas et al. (2015), Vehicle Scheduling Problem (VSP) determines the trips that are assigned to each vehicle so that all trips can be carried out in accordance with the schedule and that costs related to vehicle usage are minimized. Important elements in defining a VSP are:

- Number of depots vehicles may depart from. If there are multiple depots, it is important to define whether a vehicle must return to the same depot it departed from, or whether vehicles can share all depots.
- Number of types of fleet with specific capacities and operating costs.
- Resting points where vehicles may remain until the next trip (e.g. a street with low vehicles flow, a parking lot, and so on). A depot may be used as a resting point but a resting point does not necessarily require special infrastructure.
- Operating conditions, such as (i) inter-routing - i.e., a vehicle can be assigned to a trip of a line followed by a trip to another line - and (ii) deadhead implementation - i.e., empty vehicles traveling from a point to another, increasing the availability of vehicles at certain points where there is a greater demand. These types of operation may prevent under-usage of the fleet.

When vehicles can depart from different locations, we denoted the VSP as Multi-Depot Vehicle Scheduling Problem (MDVSP). The MDVSP is a well-known NP-hard problem (BERTOSSI; CARRARESI; GALLO, 1987). A fast heuristic approach to deal with the problem was proposed. This approach builds on two stages. The first one uses two-state space reduction procedures to reduce problem complexity. One of the procedures builds on the solutions of a single-depot vehicle scheduling for each depot, whereas the other uses the solution of a relaxed formulation of the MDVSP wherein a vehicle does not have to finish the tasks sequence in the same depot where it started. Then, the reduced problem is solved through a truncated column generation approach. The heuristic approach was implemented in several variants, through different combinations of reduction procedures, and tested on a series of benchmark problems provided by Pepin et al. (2009). The heuristic variants found solutions with very narrow gaps (below 0.7%, on average) to best-known solutions provided by Pepin et al. (2009), decreasing the required CPU time by an overall average factor of 17 in comparison with reported results in the literature (OTSUKI; AIHARA, 2014).

In the second part of this disseratation, we added complexity when considering the heterogeneous fleet, denoted as "the multiple-depot vehicle-type scheduling problem" (MDVTSP). Although several mathematical formulations and solution methods have been developed for the MDVSP, the MDVTSP is still relatively unexplored. Large instances of the MDVTSP (involving thousands of trips and several depots and vehicle types) are still difficult to solve in a reasonable time. We introduced a heuristic framework, combining time-space network, truncated column generation (TCG) and state space reduction, to solve large instances of the MDVTSP. Extensive testing was carried out using random generated instances, in which a peak demand distribution was defined based on real-world data from public transportation systems in Brazil. Furthermore, experiments were carried out with a real instance from a Brazilian city. The framework has been implemented in several algorithm variants, combining different developed preprocessing procedures, such as state space reduction and initial solutions for the TCG. Computational results showed that all developed algorithms obtained very good performances both in quality and efficiency. The best solutions, considering simultaneously quality and efficiency, were obtained in the heuristics involving state space reduction.

In the third part of this dissertation, however, we focused on the real-time reschedule for the case of serious vehicle failures. Such vehicle breakdowns require that the passengers from the disabled vehicle and those expected on the remaining part of the trip be picked up. In addition, since the disabled vehicle may have future trips assigned to it, the given schedule may be deteriorated to the extent where the fleet plan may need to be adjusted in real-time depending on the current state of what is certainly a dynamic system. Usually, without the help of a rescheduling algorithm, the dispatcher either cancels the trips that are initially scheduled to be covered by the disabled vehicle (when there are upcoming future trips planned that could soon serve the expected demand for the canceled trips), or simply dispatches an available vehicle from a depot. In both cases, there may be considerable delays introduced. This manual approach may result in a poor solution. The implementation of new technologies (e.g., automatic vehicle locaters, the global positioning system, geographical information systems, and wireless communication) in public transit systems makes it possible to implement real-time vehicle rescheduling algorithms at low cost.

Part I

Simple and efficient heuristic approach for the multiple-depot vehicle scheduling problem

Abstract

In this paper, a fast heuristic approach is proposed for solving the multiple depot vehicle scheduling problem (MDVSP), a well-known NP-hard problem. The heuristic is based on a two stage procedure. The first one applies two state space reduction procedures towards reducing the problem complexity. One procedure is based on the solutions of the single-depot vehicle scheduling for each depot, while the other uses the solution of a relaxed formulation of the MDVSP, in which a vehicle can finish its task sequence in a different depot from where it started. Next, the reduced problem is solved by employing a truncated column generation approach. The heuristic approach has been implemented in several variants, through different combinations of the reduction procedures, and tested on a series of benchmark problems provided by Pepin et al. (2009). The heuristic variants found solutions with very narrow gaps (below 0.7%, on average) to best-known solutions (PEPIN et al., 2009), decreasing the required CPU time by an overall average factor of 17 in comparison with reported results in the literature (OTSUKI; AIHARA, 2014).

Keywords: Heuristics, vehicle scheduling, multi-depots, column generation.

Note: This article has been published in the Optimization Letters. (GUEDES et al., 2015)

1 Introduction

The multi-depot vehicle scheduling problem (MDVSP) is a classical problem in operations research, arising in applications such as public transport systems, and appearing as a subproblem in the more complex crew scheduling and management disruption problems (HUISMAN; FRELING; WAGELMANS, 2005). The MDVSP has been shown by Bertossi, Carraresi e Gallo (1987) to be a NP-hard problem.

Initially, the problem was solved using heuristic methods, given the complexity of the problem and the computational power of the seventies and eighties (DESAULNIERS; HICKMAN, 2007). Carpaneto et al. (1989) developed the first optimization method, employing an “additive lower bounding" scheme. Bertossi, Carraresi e Gallo (1987) employed Lagrangean relaxation, based on a multicommodity formulation. Column generation (CG) was also used to solve the MDVSP (RIBEIRO; SOUMIS, 1994; HADJAR; MARCOTTE; SOUMIS, 2006; OUKIL et al., 2007). Löbel (1998) combined Lagrangian relaxation and CG to solve large instances based on data from three public transportation companies in Germany. Kliewer, Mellouli e Suhl (2006) introduced a new vehicle scheduling network called time-space network to solve the MDVSP. The use of this network led to a reduction in the associated mathematical models, allowing the solution of large instances by direct application of commercial integer programming solvers. Metaheuristics were also employed to solve the MDVSP. Pepin et al. (2009) developed two metaheuristic algorithms based on large neighborhood search (LNS) and tabu search, while Laurent e Hao (2009) presented an iterated local search (ILS) heuristic. More recently, Otsuki e Aihara (2014) developed a variable depth search framework which utilizes pruning and deepening techniques to speedup the CPU time required to find a good solution.

Concerning the quality and efficiency of the solutions for the problem, Pepin et al. (2009) analyzed the performance of five different approaches to solve the MDVSP, namely: truncated branch-and-cut, Lagrangian relaxation, column generation, LNS, and tabu search. The comparison showed that column generation is the best method when powerful computational resources are available, while LNS is the best option in the presence of limited resources (this result was recently ratified by Otsuki e Aihara (2014)). However, the most important conclusion of Pepin et al. (2009)’s paper is the difficulty of these heuristics to find good solution in a reasonable time for large instances. Given a time limit of one hour, the methods could only find solution for instances up to 1500 tasks and eight depots. Löbel (1998) and Kliewer, Mellouli e Suhl (2006) succeeded to solve real-world public transit instances to optimality involving up to 20000 and 7000 tasks, respectively. These instances, however, have a particular structure that ease their solution

process (PEPIN et al., 2009), being previously validated in practice.

This paper describes a simple and fast heuristic procedure to solve efficiently very large instances of the MDVSP. The heuristics consist of two sequential steps. The first step is based on a state space selection process intended to reduce the number of variables in the problem. The second step employs a CG approach to solve the reduced problem to find good solutions very quickly. The performance of variants of the proposed heuristics were assessed on the set of testbed instances by Pepin et al. (2009). All variants found very good solutions, with average gaps from the best-known solutions below 0.7 %, requiring, on average, 18 times less CPU usage. The major contributions in this paper are as follows: (i) the introduction of effective state space reduction techniques to decrease the problem complexity; and (ii) the development of a heuristic approach, based on a truncated CG approach, capable of solving very efficiently and effectively the MDVSP.

The document is organized as follows. In the next section, we present the problem and the formulations related to our solution method. Section 3 describes the development and the features of the heuristic procedure.The computational experiments and comparison with previous results are presented in Section 4. Finally, Section 5 presents some final remarks.

2 Problem

Before defining a formal mathematical formulation for the MDVSP, we introduce some useful notation. We define the movement of empty vehicles as a deadheading trip. We are given a set of N service trips, each trip $i \in N$ starting at time st_i and ending at time et_i, along with a set of K depots, in which v_k vehicles are stationed. Let t_{ij} be the deadheading transportation time between the ending point of trip i and the starting point of trip j. An ordered pair of trips (i, j) is said to be compatible iff satisfies the relation $et_i + t_{ij} \leq st_j$. The MDVSP objective is to find the minimal cost fleet schedule for executing all the service trips. A vehicle schedule is defined as a feasible sequence of service trips to perform, using the same depot as the starting and ending point.

There are several existing mathematical formulations for this problem (PEPIN et al., 2009). It is beyond the scope of this article to discuss them in detail. We briefly present the set partitioning formulation, as introduced by Ribeiro e Soumis (1994). Define the MDVSP network $G^k = \langle V^k, A^k \rangle$ for each depot $k \in K$, where V^k is the set of nodes, and A^k denotes the set of arcs. Set $V^k = \{o(k), d(k)\} \cup N$ contains a node for each trip $i \in N$ and a pair of nodes, $o(k)$ and $d(k)$, representing the depot k as the initial and final nodes of the allocated schedule of vehicle v_k. Set $A^k = \{(o(k) \times N) \cup (N \times d(k)) \cup E\}$ is the set of deadheading trips, where $(N \times d(k))$ is the set of pull-in arcs to depot k, and $E = \{(i, j) | (i, j) \quad \text{is a compatible pair of trips}, i, j \in N\}$, $(o(k) \times N)$ is the set of pull-out arcs from depot k. Let c_{ij} be the cost of transversing arc $(i, j) \in A^k$, which is dependent on traveling and waiting costs. A path from $o(k)$ to $d(k)$ represents a feasible scheduling for a vehicle in depot k. The MDVSP network can be represented by its adjacency matrix $\mathbf{MDVSP}_{|K|+|N| \times |K|+|N|}$, with $MDVSP[i][j] = c_{ij}$ if $(i, j) \in A^k$ is a compatible pair of trips, and $MDVSP[i][j] = -1$ otherwise.

Define Ω^k as the set of all feasible set of paths in $G^k, k \in K$. Let $p \in \Omega^k$ be a feasible schedule with cost c_p. For each path $p \in \Omega^k$, $a_{ip} = 1$ iff node $i \in N$ is visited in path p, $a_{ip} = 0$ otherwise. Using binary variable θ_p, with $\theta_p = 1$ if schedule p is in the solution, $\theta_p = 0$ otherwise, the MDVSP can be formulated as a set partitioning type problem as follows:

$$\min \sum_{k \in K} \sum_{p \in \Omega^k} c_p \theta_p \tag{2.1}$$

st

$$\sum_{k \in K} \sum_{p \in \Omega^k} a_{ip} \theta_p = 1 \qquad \forall i \in N \tag{2.2}$$

$$\sum_{p \in \Omega^k} \theta_p \leq \upsilon_k \qquad \forall k \in K \tag{2.3}$$

$$\theta_p \in \{0, 1\} \qquad \forall p \in \Omega^k, k \in K \tag{2.4}$$

The objective function (2.1) seeks to minimize the total costs involved. Constraints (2.2) assure that each trip $i \in N$ is visited by only one schedule $p \in \Omega^k$, while constraints (2.3) ensure that the capacity of each depot is respected. Constraints (2.4) define the domain of the decision variable.

3 Method

The framework of the proposed heuristics can be described as follows:

Step 1: State space reduction of the problem by applying:

- **Step 1.1:** k-Single Depot VSP (SDVSP), $k = 1, \ldots, K$, based selection procedure (Selection R1);
- **Step 1.2:** Relaxed-MDVSP selection procedure (Selection R2);

Step 2: Solution of the reduced problem by employing a modified truncated CG procedure.

The method is basically a two-step approach. The first step is a state space reduction method intended to reduce the set of variables to a smaller, but relevant, subset of variables, decreasing the complexity of the instances. The second step solves the reduced state space problem using an improved truncated column generation approach towards accelerating the CG stabilization, including the use of an initialization procedure based on the solutions of the $|K|$-SDVSPs.

3.1 State Space Reduction

In VSPs, it is natural that some trips are geographically more distant from others. In addition to the geographical issue, there are also incompatibility issues. This means that trips that are close geographically can be far apart due to the time in which they must start, while others may be compatible in terms of timing, but geographically infeasible. Several variables in the problem become too costly to be considered as viable alternatives in an optimal or close to optimal solution, both due to distance or timing issues. As a consequence, although the complete space is very large, the set of "relevant" variables (with high chance of being in the final solution) is much smaller. In fact, very few variables from the complete space state will be in the final solution. Rooted on this observation, we developed two selection procedures, which main objective is to identify a smaller, but representative, solution space state. Pull-in and pull-out arcs cannot be reduced, since this led to the CG solution process to become unstable, not finding feasible solutions for some problem instances. Next, the selection procedures developed are discussed in details.

3.1.1 k-SDVSP Based Selection Procedure (Selection R1)

The goal of this procedure is to identify “relevant" variables that are selected by any solution of all $|K|$-SDVSPs. The reasoning behind this procedure is quite intuitive. If a variable is not chosen as a solution considering $|K|$-SDVSPs, then it would have small chance to be chosen as a candidate solution when considering the MDVSP. Based on this reasoning, this selection procedure consists in efficiently solving $|K|$ individual SDVSPs. The arcs found in any of the solved SDVSPs configures the reduced connection graph, represented by matrix $\mathbf{MDVSP}_r$, with a similar structure to matrix **MDVSP**, in which $MDVSP_r[i][j] = MDVSP[i][j]$ if arc (i, j) is in the solution of any k-SDVSP, $k \in K$; and $MDVSP_r[i][j] = -1$, otherwise. Matrix VSP_k is easily obtained from **MDVSP** by taking into consideration only the pull-in and pull-out arcs to and from depot k, respectively. This structure enables to solve each SDVSP as an assignment problem (AP) following Paixão e Branco (1987). There are several specially designed algorithms to solve linear AP. Based on results reported by Dell'Amico e Toth (2000), the algorithm LAPJV developed by Jonker e Volgenant (1987) was employed to solve each SDVSP. Algorithm 4 outlines this selection procedure. The LAPJV algorithm has a complexity of $O(n^2.\log n)$. As Algorithm 4 is executed $|K|$ times, the complexity of the selection procedure is $O(|K|n^2.\log n)$.

Algorithm 1 k-SDVSP Based Selection Procedure

$MDVSP_r[i][j] \leftarrow MDVSP[i][j], i = 1, \ldots, |K|, j = 1, \ldots, |K| + |N|$
$MDVSP_r[i][j] \leftarrow MDVSP[i][j], i = |K| + 1, \ldots, |K| + |N|, j = 1, \ldots, |K|$
for all $k \in K$ **do**
  Generate matrix $\mathbf{SDVSP}_k$ from matrix **MDVSP**
  $Solution \leftarrow$ LAPJV($\mathbf{SDVSP}_k$)
  if arc $(i, j) \in Solution$ **then**
    $MDVSP_r[i][j] \leftarrow MDVSP[i][j]$
  end if
end for

3.1.2 Relaxed-MDVSP Selection Procedure (Selection R2)

This procedure uses the solution of a relaxed formulation of the MDVSP, relaxed-MDVSP, as the selection criterion of the variables to be included in the reduced state space. Basically, the Relaxed-MDVSP is a relaxation of the multicommodity formulation of the MDVSP (Löbel, 1998), in which vehicles are allowed to end its sequence of trips in different depots from where they started from. In order to present the formulation of the Relaxed-MDVSP, it is necessary to define the associated underlying network. Let $G = \langle V, A\rangle$ be a digraph, where $V = 1, \ldots, n$ is the vertex set containing a node for each service trip $i \in N$, and A is the set of arcs representing the deadheading trips between compatible service trips. Let $G^* = \langle V^*, A^*\rangle$ be the graph with nodes in $V^* = V \cup K$, and arcs in $A^* = A \cup A_1 \cup A_2$, where $A_1 = \{(i, j)|i \in K, j \in V\}$ represents the set of

pull-out arcs from depots, and $A_2 = \{(i,j)|i \in V, j \in K\}$ the set of pull-in arcs to depots. Introducing the decision variable x_{ij}, representing the flow in arc (i,j), the formulation for the Relaxed-MDVSP is as follows:

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij}$$

st

$$\sum_{j:(i,j)\in A^*} x_{ij} = 1 \qquad \forall i \in V \tag{3.1}$$

$$\sum_{j:(k,j)\in A_1} x_{kj} \leq v_k \qquad \forall k \in K \tag{3.2}$$

$$\sum_{i:(i,k)\in A_2} x_{kj} \leq v_k \qquad \forall k \in K \tag{3.3}$$

$$\sum_{j:(j,i)\in A^*} x_{ji} - \sum_{j:(i,j)\in A^*} x_{ij} = 0 \qquad \forall i \in V \tag{3.4}$$

$$x_{ij} \geq 0 \qquad \forall (i,j) \in A^* \tag{3.5}$$

The objective of this formulation is to minimize the total deadheading costs. Constraint (3.1) ensure that each task is executed exactly once by a vehicle. Constraints (3.2) and (3.3) limit the number of vehicles that can be used from each depot, while constraints (3.4) are flow conservation constraints which define a multiple-path structure for each depot. Constraint (3.5) defines the range of the decision variable. This problem can be seen as a minimum cost flow problem. Since all capacities and demands are integral in the Relaxed-MDVSP, the solution of this problem is composed of integer variables (WOLSEY, 1998).

The selection procedure based on the solution of the Relaxed-MDVSP is very similar to Algorithm 4, in which the Relaxed Model is solved just once by a commercial linear programming solver. If arc (i,j) is in the solution of the Relaxed-MDVSP then $MDVSP_r[i][j] = MDVSP[i][j]$, $MDVSP_r[i][j] = -1$ otherwise.

3.2 Modified Truncated Column Generation

Column generation is a well-known method to solve MDVSP (RIBEIRO; SOUMIS, 1994; OUKIL et al., 2007). As the number of paths is huge, they are generated dynamically based on a Dantzig-Wolfe decomposition, in which the restricted master problem (RMP) is a linear relaxation of model (2.1)-(2.4) and the subproblems are the shortest path problems on the network $G^k, k \in K$. Dual variables π_i and β_k are associated with constrains (2.2) and (2.3), respectively. At iteration t, the modified costs of arc $(i,j) \in A^k$ is computed by $c_{ij} - \pi_i^t$, if $i \in N$, and $c_{ij} - \beta_i^t$, if $i = o(k)$. The RMP is solved for each iteration to a subset of variables θ_p. The dual variables of the RMP are used in the subproblems to both

test the optimal solution (if the reduced costs of all variables are all non-negative) and to generate new columns (paths found in the subproblems which reduced cost is negative).

Our CG approach toward solving the reduced MDVSP problem is based on the truncated CG algorithm described in Pepin et al. (2009). The algorithm requires three predefined parameters, namely Z_{min}, I, and Ω_{min}. The algorithm terminates early if the optimal value of the RMP has not decreased by more than Z_{min} in the last I iterations. Parameter Ω_{min} is a threshold value in the rounding of variables θ_p toward an integer solution. However, we introduced some changes to this algorithm. The modified algorithm uses two different (but similar) formulations to solve the reduced MDVSP problem. One problem is the traditional RMP as presented by Ribeiro e Soumis (1994). The second formulation, called the relaxed restricted master problem (RRMP), replace constraints (2.2) by the following constraints:

$$\sum_{k \in K} \sum_{p \in \Omega^k} a_{ip} \theta_p \geq 1 \qquad \forall i \in T \tag{3.6}$$

keeping the same objective function and remaining constraints. These two problems have different dual solutions. The rationale behind this change is to restrict the dual variables in sign (LÜBBECKE; DESROSIERS, 2006). Let π_i and $\pi_i^{'}, \forall i \in N$ be the dual variables of constraints (2.2) and (3.6), respectively. In the standard CG procedure, the optimal solution of the RMP is obtained when the reduced costs of the corresponding path variables are non-negative. This cost is computed by the expression $c_{ij} - \pi_i$ at each iteration of the method. Since π_i can assume negative values given the equality in constraints (2.2), the reduced cost may be positive for several iterations. As a consequence, the convergence of the method can be rather slow, since the RMP objective function can decrease very slowly. As $\pi_i^{'}$ can only assume positive values, we expect that the reduced costs of paths p obtained in the subproblems, $c_{ij} - \pi_i^{'}$, will be often disturbed when compared with the standard CG. Better bounds are often found, since the dual solutions are more frequently changed. Since the solution of the RRMP does not respect all constraints in the MDVSP, this model can be used until the problem becomes "stagnated", e.g., when the subproblems were not able to generate new columns, since all reduced costs are positive. At this stage, we change the RRMP to the standard RMP. The solution process continues until an integer solution of variables θ_p is found.

Another change introduced was in the way the columns are inserted in the master problem. The algorithm described by (PEPIN et al., 2009) insert at most $|K|$ columns at each iteration of the CG procedure, depending on the sign of the reduced cost of each $k \in K$ subproblem. The master problem, with these new columns, is solved once at the beginning of the next iteration of the CG procedure. Our approach uses the traditional way of selecting columns to insert in the current master problem. However, it solves

the master problem right after a new column is inserted. During experimentation, we noticed that the inclusion of several columns at each instance was generating many similar columns, unnecessarily solving several subproblems. This problem was observed in several CG applications (LÜBBECKE; DESROSIERS, 2006). Although this procedures solves initially a higher number of master problems, the use of updated duals helps to stabilize the CG, resulting in better convergence.

4 Computational Results

The performance of the heuristic approach was evaluated in a set of instances available at the public site http://people.few.eur.nl/huisman/instances.htm, referred to as Huisman's website. The instances comprise problems sizes with 4 and 8 depots and 500, 1000, and 1500 trips, totaling 30 test problems. The number of depots (m), the number of trips (n) and the instance id (s0, s1, s2, s3, and s4) are indicated in the name of the instance. We have set the fixed costs of the depots pull-in and pull-out arcs high enough (a total of 10000) to allow the minimum cost flow problem determine the optimal number of vehicles required for the whole journey, following Pepin et al. (2009). The proposed heuristics were implemented in C++ and used CPLEX 12.5 to solve the linear programming models involved in the CG and in selection R2. All experiments were performed on a computer Intel Core I5-3210 processor running at 2.80 gigahertz under Linux kernel 3.12 (64 bits) with 8 gigabytes of RAM.

CG requires an initial solution. Initially, we replicated the use of artificial variables penalized by a big-M cost as defined in Pepin et al. (2009). However, we faced a solution process that led to poor convergence and very high CPU times for all instances. CG required excessive time to find solutions without artificial variables in the basic solution of the RRMP. Although a good initial solution cannot guarantee a good convergence process (LÜBBECKE; DESROSIERS, 2006), we decided to use the paths obtained by Algorithm 4 as an initial set of columns to the RRMP, regardless of applying Selection R1. Its primal optimal solution is used to compute an initial solution for the MDVSP, offering an upper bound on the integer optimal value. Its dual solution provides a lower bound on the RRMP.

The following notation is used to indicate the implemented algorithms: (i) MTCG - the modified truncated column generation using the initialization routine (ii) R1 - state space reduction of the problem by applying only selection R1; (iii) R2 - state space reduction of the problem by applying only selection R2; (iv) R1+R2 - state space reduction of the problem by applying selection R1 and R2. The last three algorithms are variants of the developed heuristic and use MTCG to solve the reduced space state in the second phase of the heuristic approach.

The settings of CG parameters were carried out during the experiments. The best quality results were obtained with small values of Z_{min} and large values of I and Ω_{min}. The best solutions were found with $Z_{min} = 0$. Concerning the number of minimum iterations, MTCG obtained good solutions with $I = 5$. As we reduced the state space of an instance to be solved by the CG procedure, it was necessary to increase the value of this parameter

to $I = 30$, as a compensation factor. Regarding Ω_{min}, good solutions (in terms of quality and efficiency) were obtained with this value either set to 0.8 or 0.9.

Table 1 presents a detailed comparison of the developed algorithms for the 30 instances in Huisman's website, concerning solution quality. For each algorithm, Table 1 displays the optimal solution found and the solution gap (in %). The gaps were computed using $GAP = 100 * \frac{(V-VB)}{VB}$, where V is the value computed by the evaluated method and VB is the solution value informed in Huisman's website. Average and maximum gaps (in %) in relation to Huisman's best available solutions (using CPLEX, truncated branch-and-cut, or TCG) were also reported in this table. As all developed algorithm have succeeded to find the same optimal number of vehicles as presented by Huisman's website for all instances, we decided to omit these values for economy's sake. We refer to this website for the obtained values.

MTCG obtained very similar results to the ones reported by Huisman's site, with gaps below 0.16% for all instances. These gaps could be decreased for some instances, if a loss of efficiency is tolerated, since the solutions presented in Table 1 are good trade-off values between running time and quality. Better quality solutions could be obtained by increasing CG parameters I and Ω_{min}. In terms of efficiency, MTCG also presented competitive values with the results reported by Pepin et al. (2009), on average reducing the CPU times in around 32%. This reduction was expected, considering the use of more modern computational resources. MTCG efficiency is justified by the different path structures in the subproblems of the CG procedure, resulting from the introduced changes described in Section 3.2. Given the obtained results, MTCG was considered a valid and competitive CG implementation for the MDVSP.

Table 2 compares the optimality gap of our developed algorithms with the following state-of-the-art solutions reported in the literature (PEPIN et al., 2009; LAURENT; HAO, 2009; OTSUKI; AIHARA, 2014) for each category instance: (i) Lagrange relaxation (LR); (ii) Large neighborhood search combined with CG (LNS); (iii) Tabu search (TS); (iv) Ejection chain based approach (EC); and (v) Variable depth search algorithm (VDS). The optimality gaps were computed as follows:

$$\text{Optimality GAP (in \%)} = 100 * \frac{(VA - V_{best})}{V_{best}}$$

where V_{best} is the best known solution, and VA is the solution value obtained by the algorithm being compared, say algorithm A. The values presented in this table refer to the best solution found for each instance category.

Table 3 shows the average CPU time spent for each algorithm to find the best solution values, comparing them with the CPU times reported by Pepin et al. (2009) and

Table 1 – Solution quality comparison with Huisman's results

	Huisman	MTCG		Selection R1		Selection R2		Selection R1+R2	
Instance	Solution	Solution	Gap	Solution	Gap	Solution	Gap	Solution	Gap
m4n500s0	1289114	1289280	0.01	1297940	0.68	1300900	0.91	1296600	0.58
m4n500s1	1241618	1241970	0.03	1247940	0.51	1247170	0.45	1247960	0.51
m4n500s2	1283811	1284360	0.04	1292650	0.69	1298200	1.12	1291547	0.60
m4n500s3	1258634	1259290	0.05	1267820	0.73	1267260	0.69	1266780	0.65
m4n500s4	1317077	1317310	0.02	1323870	0.52	1325750	0.66	1322490	0.41
m4n1000s0	2516247	2516580	0.01	2539920	0.94	2539430	0.92	2534440	0.72
m4n1000s1	2413393	2414020	0.03	2436200	0.95	2443230	1.24	2427680	0.59
m4n1000s2	2452905	2454390	0.06	2467880	0.61	2463880	0.45	2461690	0.36
m4n1000s3	2490812	2491950	0.05	2506060	0.61	2507770	0.68	2503240	0.50
m4n1000s4	2519191	2523100	0.16	2525980	0.27	2524680	0.22	2524420	0.21
m4n1500s0	3830912	3832930	0.05	3855030	0.63	3861470	0.80	3853490	0.59
m4n1500s1	3559176	3560920	0.05	3571280	0.34	3570820	0.33	3570820	0.33
m4n1500s2	3649757	3650790	0.03	3676580	0.73	3669560	0.54	3662380	0.35
m4n1500s3	3406815	3408510	0.05	3436530	0.87	3428770	0.64	3435040	0.83
m4n1500s4	3567122	3567740	0.02	3591240	0.68	3588810	0.61	3589170	0.62
m8n500s0	1292411	1292590	0.01	1302270	0.76	1300370	0.62	1300810	0.65
m8n500s1	1276919	1277300	0.03	1285350	0.66	1284500	0.59	1284080	0.56
m8n500s2	1304251	1304530	0.02	1310880	0.51	1309920	0.43	1309930	0.44
m8n500s3	1277838	1278030	0.02	1286140	0.65	1284550	0.53	1284590	0.53
m8n500s4	1276010	1276210	0.02	1283890	0.62	1282690	0.52	1282760	0.53
m8n1000s0	2422112	2422410	0.01	2440000	0.74	2439170	0.70	2438330	0.67
m8n1000s1	2524293	2524640	0.01	2536730	0.49	2534950	0.42	2535240	0.43
m8n1000s2	2556313	2556320	0.00	2571010	0.57	2578960	0.89	2569170	0.50
m8n1000s3	2478393	2478390	0.00	2488730	0.42	2488060	0.39	2487920	0.38
m8n1000s4	2498388	2499840	0.06	2509520	0.45	2509960	0.46	2508890	0.42
m8n1500s0	3500160	3500580	0.01	3522880	0.65	3520360	0.58	3520520	0.58
m8n1500s1	3802650	3802480	0.00	3815900	0.35	3840640	1.00	3813860	0.29
m8n1500s2	3605094	3605680	0.02	3627980	0.63	3625970	0.58	3625950	0.58
m8n1500s3	3515802	3516100	0.01	3534960	0.54	3531310	0.44	3531480	0.45
m8n1500s4	3704953	3705220	0.01	3729960	0.67	3726310	0.58	3726490	0.58
Average			0.03		0.62		0.63		0.51
Maximum			0.16		0.95		1.24		0.83

Table 2 – Solution quality comparison of the developed algorithms with other methods

	Optimality Gaps (%)								
Category	LR	LNS	TS	EC	VDS	MTCG	R1	R2	R1+R2
m4n500	3.12	2.18	10.92	1.85	1.29	0.01	0.51	0.45	0.41
m4n1000	3.88	1.41	8.31	1.22	1.05	0.01	0.27	0.45	0.21
m4n1500	5.54	2.05	10.78	1.86	1.40	0.02	0.34	0.33	0.33
m8n500	5.54	2.05	17.87	3.00	2.26	0.01	0.51	0.43	0.44
m8n1000	6.59	3.25	19.65	2.47	2.26	0.00	0.42	0.39	0.38
m8n1500	10.15	3.69	20.82	2.84	2.44	0.00	0.35	0.44	0.29

Otsuki e Aihara (2014), called as VDS in the table. It is important to notice that this efficiency comparison with the CPU times presented by Pepin et al. (2009) and Otsuki e Aihara (2014) should be viewed with extreme cautions, since we performed the experiments six years after the former research work, and using different experimental conditions of both studies. Although we employed faster hardware and softwares, the direct implementation of the truncated CG presented in Pepin et al. (2009), using C++ and CPLEX 12.5, took excessive CPU time. In order to obtain comparable CPU usage we needed to implement our MTCG with the initialization process described above. The CPU times reported by Pepin et al. (2009) were obtained using GENCOL, a computer program specially designed to solve a broad class of routing and scheduling problems. Such softwares employ special routines to substantially reduce the CPU time to solve a problem. However, the majority of these softwares are not of public domain, with a cost beyond the budget of the average transport company in development countries. As a consequence, we would like to point out that a precise comparison can only be carried out among our developed algorithms, having the same data structure and the same CPLEX parametrization. Only rough comparisons can be made using Pepin et al. (2009) and Otsuki e Aihara (2014)'s results.

Table 3 – Solution efficiency comparison of the developed algorithms with other methods

	Average CPU Time (s)					
Category	Pepin et al.(2009)	VDS	MTCG	R1	R2	R1+R2
m4n500	77	71	62	3	4	4
m4n1000	651	612	423	7	22	28
m4n1500	2203	2012	868	56	86	79
m8n500	119	109	175	6	13	8
m8n1000	857	787	1380	73	103	97
m8n1500	3085	2800	1813	182	208	199

Tables 1, 2, and 3 show that the variants of the developed heuristics, in general, offer good quality solutions (some with very tight gaps) in comparison with other methods, very efficiently. Table 2 shows that our developed algorithms present the best solution quality among the compared methods, overcoming mathematical programming methods such as LR, metaheuristic methods such as TNS and EC, and local search methods such as LNS and VDS. As expected, the solution quality improved as the reduced state space is enlarged by the combination of selection procedures. An opposite behavior is observed concerning the efficiency of the solution, characterizing a trade-off between CPU time and the optimal value of the objective function. Algorithms R1 and R2 presented similar average solution gaps for the tested instances (around 0.62%). However, R2 presented a slightly higher maximum gap. Variant R1+R2 obtained the best objective function values among the reduction state space based algorithms, with average and maximum gaps of 0.51% and 0.83%, respectively. All three variants were extremely fast in comparison with MTCG. R1 is roughly 21 times faster, on average, than MTCG; R2 is 13 time faster, on average, than MTCG; while variant R1+R2 is 14 times faster, on average, than MTCG.

Variants R1, R2, and R1+R2 were 21, 16, and 17 times quicker, on average, than the CPU times reported by Pepin et al. (2009); and 19, 14, and 15 times quicker, on average, when compared with Otsuki e Aihara (2014)'s CPU times. Moreover, this latter study claimed that VDS shows the best short-term performance, since this method obtained a good feasible solution in around 300s CPU time for category m4n1500. Variants R1, R2, and R1+R2 obtained, for the same category, near best available solutions in one quarter of this time, on average, making our developed heuristic framework a very competitive short-term performance method for solving the MDVSP.

We could not define a pattern in the CPU times reduction based on our experiments. For instance, variant R2 was on average quicker to solve instances with 4 depots, while variant R1+R2 presented an opposite behavior. The best CPU times reductions were obtained for instances with 4 depots and 1000 trips. Further experimentation will be required to explain both behaviors.

Overall, the developed heuristic approach offered very competitive algorithms to solve the MDVSP, specially variants R1 and R1+R2. Both variants obtained very good solutions, with maximum gaps below 1% with very low CPU usage in comparison to previous reported results (PEPIN et al., 2009; OTSUKI; AIHARA, 2014). Variant R1+R2 offered the best compromise solutions in terms of quality and efficiency. This variant can be a good option in contexts and where solution quality is an important criterion. However, variant R1 was a quite good surprise, finding best solution values with reasonable average solution gaps (0.63%), but requiring the minimum CPU time of all variants. We recommend its application in real-time transportation logistics environments, where the MDVSP is solved several times as a subproblem, and solution quality is an important, but no essential criterion. Variant R2 was clearly dominated by the remaining variants. Nevertheless, the final selection among these three variants is case dependent, and it will depend on further experimentation, with a special attention on the settings of CG parameters.

5 Conclusion

In this paper, we presented a two-phase heuristic to solve the MDVSP. First, the heuristics employs a combination of procedures to select, from the whole feasible solution space, a good set of arcs to compose the solution of the problem. Each procedure is based on the following selection criteria: (i) the set of paths found in the solution of $|K|$-SDVSPs; and (ii) the set of paths found in the solution of a relaxed MDVSP formulation, where the sequence of trips carried out by a vehicle can start and finish in different depots. The selected set of arcs is then solved by using a modified truncated CG algorithm. The developed approach led to very competitive method to solve the MDVSP.

Different combinations of the selection procedures were tested, resulting in three different variants. On the one hand, variant R1 achieved narrow gaps to best-known solutions (0.62% on average) with exceptional running times (roughly 34 times faster, on average, than previous reported results). On the other hand, the variant R1+R2 offered the best trade-off between solution quality and running times. It was, on average, 16 times faster than previous reported results (PEPIN et al., 2009; OTSUKI; AIHARA, 2014), obtaining solutions with an average gap of around 0.5%.

Future research is directed toward using the heuristic method to develop novel approaches for solving both the real time vehicle recovery scheduling problem and the integrated crew and VSP. In both problems, the MDVSP needs to be solved several times as a subproblem. Moreover, we intend to alter the developed method to consider the time-space network (KLIEWER; MELLOULI; SUHL, 2006) as the underlying vehicle scheduling network.

Part II

Column generation based heuristic framework for the multiple-depot vehicle type scheduling problem

Abstract

The multiple-depot vehicle-type scheduling problem (MDVTSP) is an extension of the classic multiple-depot vehicle scheduling problem (MDVSP), where heterogeneous fleet is considered. Although several mathematical formulations and solution methods have been developed for the MDVSP, the MDVTSP is still relatively unexplored. Large instances of the MDVTSP (involving thousands of trips and several depots and vehicle types) are still difficult to solve in a reasonable time. We introduce a heuristic framework, combining time-space network, truncated column generation (TCG) and state space reduction, to solve large instances of the MDVTSP. Extensive testing was carried out using random generated instances, in which a peak demand distribution was defined based on real-world data from public transportation systems in Brazil. Furthermore, experiments were carried out with a real instance from a Brazilian city. The framework has been implemented in several algorithm variants, combining different developed preprocessing procedures, such as state space reduction and initial solutions for the TCG. Computational results show that all developed algorithms obtained very good performances both in quality and efficiency. The best solutions, considering simultaneously quality and efficiency, were obtained in the heuristics involving state space reduction.

Keywords: bus scheduling, heterogeneous fleet, column generation, time-space network, state space reduction.

Note: This article has been published in the Computers & Industrial Engineering. (GUEDES; BORENSTEIN, 2015)

1 Introduction

The multiple depot vehicle-type scheduling problem (MDVTSP) consists in finding the minimum cost assignment of a set of vehicles with different technical specifications from several depots to a set of trips previously defined. The MDVTSP can be seen as a generalization of the classic multiple-depot vehicle scheduling problem (MDVSP), where heterogeneous fleet is considered. The MDVTSP arises in a wide array of practical applications. Instances of MDVTSP occur in public transportation, maritime, rail, or air transportion. Because of our initial motivation, which arose from bus scheduling, in this paper special contextual reference is made to the MDVTSP in public transit systems, where a bad scheduling planning results in the increase of operational and fixed costs.

Although the literature describes several different approaches to solve the MDVSP (PEPIN et al., 2009), the MDVTSP has not received the same attention in the literature than its counterpart (CEDER, 2011b). This is quite a surprise, since heterogeneous fleet is a common characteristic in transit networks of most large size cities around the world (ROMAN, 2012). Nevertheless, some solution methods were proposed in the literature to solve the MDVTSP, mainly based on heuristics (HASSOLD; CEDER, 2014). However, due to the huge size of the underlying vehicle scheduling network, they either solved limited size problems requiring very large amount of CPU time, or mischaracterized the problem, relaxing some of the related constraints. These limitations are restricting their application to real world transportation systems. The goal of this research is to address this gap in the literature.

This paper describes a heuristic framework based on column generation (CG) to solve very large instances of the MDVTSP with a good compromise between efficiency and quality of the solution. The whole heuristic framework consists of four sequential steps. In the first step, the vehicle underlying network is generated both as a time space network (TSN) and as a connection network. The second step is based on a state space reduction process intended to reduce the number of variables in the problem. In the third step, initial solutions for the next phase are generated, based on the developed reduction process of the second phase. Finally, the fourth step employs a CG approach to solve the original or the reduced problem to find near optimal solutions. Several algorithm variants were tested, combining these steps in a plug-and-play approach. The performance of these variants was assessed on randomly generated instances up to 3000 trips, 32 depots, and 8 vehicle types. Computational comparisons showed that some of the variants achieved very good results, requiring very competitive CPU time, to solve all tested instances. The developed algorithms constitute a viable alternative to solve efficiently the MDVTSP.

The contributions of this paper are as follows: (i) to introduce an efficient and effective state space reduction technique to decrease the MDVTSP complexity; (iii) to develop a costomized truncated CG approach towards solving the MDVTSP with efficency; (iii) to develop an integrated heuristic framework, combining several modeling and solution techniques, to accelerate the solution process; and (iv) to systematically analyze the performance of the developed algorithms that compose the heuristic framework.

The paper is organized as follows. Section 2 reviews the literature on the MDVTSP. Section 3 presents the develped mathematical formulation of the problem. The developed branch and price framework for solving the problem is presented in Section 5. In Section 5.5, a reduced state space procedure is proposed to further reduce computation times. Section 6 describes the computational experiments carried out to evaluate the performance of the algorithms. Finally, a summary of the results, and areas of future research are provided in Section 7.

2 Literature Review

The literature on MDVTSP is still scarce, possibly reflecting the difficulty of solving the problem. In the MDVTSP, the number of possible vehicles routes, considering different depots and vehicle types, results in a combinatorial complexity for the problem. The MDVTSP was modeled and solved using similar methods than the ones developed for the MDVSP. Gintner, Kliewer e Suhl (2005) and Kliewer, Mellouli e Suhl (2006) developed heuristic methods to solve the MDVTSP, which main idea was initially to solve simplified models for each depot, and then search for commons chains in the solutions. If a common chain is included in a vehicle scheduling problem (SDVSP) solution, the authors classify this as a stable chain and assume that may occur in the global optimal solution. The stable chains were treated as the only trips, reducing the complexity of the MDVSP, in a direct application of standard optimization software. However, significant simplifications were made, mischaracterizing the problem as the classic MDVTSP, as follows: (i) one-depot-per-route requirement was relaxed, allowing a vehicle to start and finish its sequence of tasks in different depots; and (ii) each vehicle type was assigned to only one depot. With these simplifications, the authors solved instances up to 7068 trips and 5 depots, requiring 3 hours of CPU time. van den Heuvel, van den Akker e Niekerk (2008) extended Gintner, Kliewer e Suhl (2005)'s formulation, explicitly adding the vehicle-type in the new formulation. In addition, passenger demand was also considered. However, to obtain a solution in a reasonable time, the authors considered only one depot and allowed more than one vehicle of the same type to perform a trip.

Some authors used alternative approaches to model and solve the MDVTSP. Laurent e Hao (2009) considered the heterogeneous fleet within the MDVSP, formulated as a graph coloring problem. An iterative tabu search method was employed to minimize the required number of vehicles. They tested the developed method in seven real instances with eight vehicle types. Good results in terms of solution quality and low computational times were obtained in the experiments. Oughalime et al. (2009) presented theoretical efforts to perform the integration of vehicle scheduling with heterogeneous fleet and crew scheduling. The authors proposed a sequential modeling that uses integer programming to solve the vehicle scheduling problem and a goal programming model for the crew scheduling. Although the research is constructed using a real example, no experiments were presented to validate the proposed model. Ramos, Reis e Pedrosa (2011) modeled the MDVTSP as an asymmetric traveling salesman problem (ATSP). They proposed an ant colony heuristic method to solve the problem. However, as the ATSP is a well-known NP-complete problem, the developed approach presented some difficulties in finding good solutions in a reasonable time.

Ceder (2011b) used an innovative approach to solve the integration of the vehicle scheduling considering heterogeneous fleet, called Deficit-Function Theory (DFT). As defined in (CEDER, 2011b), the DFT is a heuristic method that uses the deficient number of vehicles in a particular terminal in a multi-terminal transport public system to solve the problem. The MDVTSP is formulated as a minimum cost network flow and solved using a heuristic procedure that allows flexible departure times of trips within a tolerance interval, following rules expressed in the DFT. The method was used in two small examples. Ceder (2011a) improved his heuristic framework, applying it to several sets of real data from the transit system of Auckland, New Zealand. Recently, (HASSOLD; CEDER, 2014) proposed a multi-objective model for the vehicle scheduling and timetabling generation integrated problem. The model allowed to stipulate the use of a particular vehicle type for a trip or to allow for a substitution either by a larger vehicle or a combination of smaller vehicles with the same or higher total capacity. Moreover, a variant in the method made it possible to construct sub-optimal timetables given a reduction in the vehicle scheduling cost. It was demonstrated that a substitution of vehicles is beneficial and can lead to significant cost reductions. However, the method can only be applied to individual bus lines, not considering interconnected lines or a set of lines, and therefore with limited applicability to the MDVTSP

Although the literature has interesting and useful ideas towards the modeling and solution of the MDVTSP, most of the available algorithms are computationally intensive, requiring thousands of CPU seconds to solve real-world scheduling instances, involving thousands of trips, and dozens of vehicles and depots. Moreover, the quickest developed approaches relax some of the MDVTSP constraints and requisites, making it easier to find solutions. However, there is either limited notion of the quality of the obtained solutions, or their behavior in practical applications in real situations. As a consequence, it is unclear whether they can be directly applied to large instances of the classic MDVTSP problem.

3 Problem Definition and Formulation

The MDVTSP can be defined as follows. Given a set of vehicle types with significantly different attributes; a set of depots with their vehicle type capacity; a set of trips with fixed peak demand, and starting and ending times; given the travel times between all pair of locations, find a feasible minimum-cost scheduling in which (i) each vehicle performs a feasible sequence of trips, starting and ending in the same depot; (ii) the peak demand of each trip is always attended by a vehicle; and (iii) the capacity of each depot for each vehicle type is respected. Before defining a mathematical formulation for the problem, we first introduce the vehicle scheduling network.

3.1 Vehicle Scheduling Network

In order to represent the scheduling underlying network, a time-space network (TSN) structure was chosen rather than the more traditional connection network (RIBEIRO; SOUMIS, 1994). On a TSN, the nodes represent a specific location in time and space, and each arc corresponds to a transition in time and possibly space. TSN was first proposed for routing problems in air scheduling (HANE et al., 1995), due to its ease in modeling possible connections between flights. Kliewer, Mellouli e Suhl (2006) introduced TSN in the context of vehicles scheduling problems.

The TSN formal definition needs some notation. We are given a set of ST service trips. Let S be the set of terminals or stations; K be the set of depots and T the planning time horizon. Each trip $i \in ST$ starts at time $st_i \in T$ and in terminal $ss_i \in S$, and ends at time $et_i \in T$ and in terminal $es_i \in S$. Let $o(k) \in S$ and $d(k) \in S$ denote the same depot $k \in K$, where $o(k)$ means the depot as a vehicle' starting point, and $d(k)$ as its terminating point. Define $N = \{(l,t) \,|\, l \in S, t \in T\}$ as the set of nodes in the TSN. Arcs are defined formally as follows.

Let $A_{se} = \{(i_s, i_e) \in N \times N \,|\, i_s = (ss_i, st_i), i_e = (es_i, et_i)\}$ be the set of service trips, in which vehicles are transporting passengers.

Let $A_{wait} = \{(i_e, j_s) \in N \times N \,|\, i_e = (es_i, et_i), j_s = (ss_j, st_j), es_i = ss_j, st_j > et_i\}$ be the set of waiting trips, representing the transitions between the end of a trip i, represented by node $i_e \in nN$ and the beginning of trip j, represented by node $j_s \in N$. A vehicle assigned to a waiting trip is stopped in a terminal.

Let $A_{dh} = \{(i_e, j_s) \in N \times N \,|\, i_e = (es_i, et_i), j_s = (ss_j, st_j), es_i \neq ss_j, st_j \geq et_i + \gamma_{ij}\}$ be the set of deadheading arcs, where γ_{ij} is the transportation time between the ending

terminal of trip i and the starting terminal of trip j. Trips i and j are called a compatible pair of trips, since the same vehicle can be assigned to both trips. A deadheading trip is a movement of an empty vehicle to a destination without picking up or dropping off passengers.

Let $A_{pout} = \{(n_1, i_s) \in \{o(k)\} \times N \,|\, n_1 = (o(k), t), i_s = (ss_i, st_i), k \in K\}$ be the set of pull-out arcs that represent the deadheading trips from depot to some terminal in order to finish a sequence of service trips, while set $A_{pin} = \{(i_e, n_2) \in N \times \{d(k)\} | i_e = (es_i, et_i), n_2 = (d(k), t), k \in K\}$ denotes the set of pull-in arcs that represent the deadheading trips from some terminal to depot, starting a sequence of service trips by a vehicle.

Let $A_c = \{(n_1, n_2) \in N \times N \,|\, n_1 = (o(k), t_1), n_2 = (d(k), t_2), k \in K, t_2 > t_1\}$ be the set of circulation arcs, representing the flow from the last to the first depot node. This arc is an artificial one, and serves the purpose of counting the number of performed vehicles blocks.

The TSN can be defined as a directed graph (N, A), with nodes N as previously defined, and arcs $A = A_{se} \cup A_{wait} \cup A_{dh} \cup A_{pin} \cup A_{pout} \cup A_c$. A vehicle route is a sequence of trips in which all consecutive pairs of trips are compatible. It can be represented in a TSN as a sequence of arcs $a_1, a_2, \ldots, a_{n-1}, a_n, a_i \in A, \forall i$, in which $a_1 \in A_{pin}, a_{n-1} \in A_{pout}, a_n \in A_c$. The main advantage of the TSN is to offer additional information in comparison with the connection network that allows to reduce substantially the number of deadheading arcs (STEINZEN et al., 2010). Kliewer, Mellouli e Suhl (2006) and van den Heuvel, van den Akker e Niekerk (2008) developed specially design procedures to reduce the size of the TSN, taking advantage of the explicit representation of time in this network. We refer to these two papers for a detailed description of these efficient reduction processes. If the problem contains m terminals and n trips, the number of deadheading arcs on a TSN becomes $O\big(mn\big)$, in comparison to $O\big(n^2\big)$ of the connection network, with $n >> m$. TSN is especially relevant when the number of terminals involved in the problem is low compared to the number of trips.

3.2 Mathematical Formulation

Given a TSN, $G^{kf} = (N^{kf}, A^{kf})$, for each depot $k \in K$ and for each vehicle type $f \in F$, the MDVTSP can be formulated as an integer linear programing model. In the MDVTSP context, an arc in the deadheading set should respect not only time compatibilities, but the capacity of a vehicle, e.g., the peak demand of trips i and j must be smaller than the capacity of the vehicle assigned to perform both trips. We first introduce the parameters. Let c_{ij}^{kf} be the cost of a vehicle type f from depot k being assigned to arc $(i, j) \in A^{kf}$. Let d_i be the peak demand of trip i. Let q_f be the capacity of vehicle f. Let v_f^k be the number of vehicles type f in depot k. We now introduce the decision variables.

Let x_{ij}^{kf} be a binary decision variable, with $x_{ij}^{kf} = 1$ if a vehicle type f from depot k is assigned to trip j directly after trip i, and $x_{ij}^{kf} = 0$ otherwise. The MDVTSP is formulated as follows:

$$\min \sum_{k \in K} \sum_{f \in F} \sum_{(i,j) \in A^{kf}} c_{ij}^{f} x_{ij}^{kf} \tag{3.1}$$

s.t.

$$\sum_{f \in F} \sum_{k \in K} \sum_{j \in (i,j) \in A^{kf}} q_f x_{ij}^{kf} \geq d_i \qquad \forall i \in A_{se}^{kf} \tag{3.2}$$

$$\sum_{f \in F} \sum_{k \in K} \sum_{j \in (i,j) \in A^{kf}} x_{ij}^{kf} = 1 \qquad \forall i \in A_{se}^{kf} \tag{3.3}$$

$$\sum_{j:(o(k),j) \in A^{kf}} x_{i,j}^{kf} \leq v_f^k \qquad \forall k \in K, f \in F \tag{3.4}$$

$$\sum_{j:(j,i) \in A^{kf}} x_{ji}^{kf} - \sum_{j:(i,j) \in A^{kf}} x_{ij}^{kf} = 0 \qquad \forall i \in L^{kf}, \forall k \in K, f \in F \tag{3.5}$$

$$x_{ij}^{kf} \in \{0,1\} \qquad \forall (i,j) \in A^{kf}, k \in K, f \in F. \tag{3.6}$$

where $L^{kf} = \{(l,t) | l \in S - \{o(k)\}, \{d(k)\}, t \in T\}$. Clearly, $L^{kf} \subset N^{kf}$.

The objective function (4.1) minimizes the total operational costs. Constraints (4.2) ensure that the peak demand of every trip is respected. The constraints (4.3) ensure that each trip is executed exactly once for a vehicle. Constraints (4.4) limit the number of vehicles that can be used from every depot, while (4.5) are flow conservation constraints. The domain of the decision variables is defined by constraints (4.6).

If we consider $f = 1$, and remove constraints (4.2), we obtain the classic multi-commodity formulation of the MDVSP. Bertossi, Carraresi e Gallo (1987) proved that the MDVSP is a NP-hard problem. Therefore, the MDVTSP is also NP-hard.

4 Algorithms

In this section, we propose CG based algorithms for solving the MDVTSP. Firstly, we describe a branch-and-price framework, using a truncated CG algorithm. Next, we introduce a state space reduction preprocessing routine designed to speedup the solution process for medium- and large-sized instances.

4.1 Truncated Column Generation

In the MDVTSP, the number of possible vehicles routes, considering the different depots and vehicle types, results in a very large number of variables and constraints. The reformulation of the problem and the use of Dantzig-Wolfe decomposition can effectively help in solving the MDVTSP with a good compromise between efficiency and solution quality.

Rather than developing a method for generating a specialized CG algorithm based on the TSN, we developed a procedure to convert a TSN into a connection network, taking advantage of the CG approaches described in previous research works for the MDVSP (RIBEIRO; SOUMIS, 1994; PEPIN et al., 2009). A connection network for the MDVTSP is an acyclic digraph (V^{kf}, E^{kf}), $k \in K$, $f \in F$ with nodes $V^{kf} = \{o(k), d(k)\} \cup ST$, and arcs $E^{kf} = \{E^f \cup (o(k) \times ST) \cup (ST \times d(k)\}$, where $E^f = \{(i,j) | st_j \geq et_i + \gamma_{ij}, q_f \geq d_i, q_f \geq d_j, i, j \in ST\}$ is the set of deadheading trips, $(o(k) \times ST)$ is the set of pull-out arcs from depot k, and $(ST \times d(k))$ is the set of pull-in arcs to depot k. Algorithm 2 describes the developed routine.

A connection network for the MDVTSP can be represented by its adjacent matrix $\mathbf{MDVTSP}^{kf}$, a matrix of costs $(n+m) \times (n+m)$ where n is the number of trips and m is the number of depots. $MDVTSP^{kf}[i][j] = c_{ij}^{kf}$, if $(i,j) \in E^{kf}$, and $MDVTSP^{kf}[i][j] = -1$ otherwise.

4.1.1 Primal Problem

The MDVTSP can also be formulated as a set partitioning problem, based on the formulation of (RIBEIRO; SOUMIS, 1994) for the MDVSP. Let $G^{kf} = (V^{kf}, E^{kf})$ be a connection network corresponding to a depot $k \in K$ and a vehicle type $f \in F$. Let Ω^{kf} be the set of all possible paths carried out by vehicle $f \in F$ from $o(k)$ to $d(k), k \in K$. Let p be a vehicle route or path in the vehicle scheduling network. Let c_p the cost of executing route p. Let $a_{ip} = 1$ if trip $i \in V^{kf}$ is covered by route p, and $a_{ip} = 0$ otherwise. Let binary

Algorithm 2 TSN to connection network conversion routine

for all $k \in K$ **do**
  for all $f \in F$ **do**
    Set $V^{kf} \leftarrow \emptyset, E^{kf} \leftarrow \emptyset$
    for all $((i_e, j_s) \in A^{kf}_{wait} \cup A^{kf}_{dh})$ **do**
      $V^{kf} \leftarrow V^{kf} \cup \{i\} \cup \{j\}$
      $E^{kf} \leftarrow E^{kf} \cup \{(i,j)\}$
    end for
    for all $((n, i_s) \in A^{kf}_{pout} | n = (o(k), t))$ **do**
      if $(o(k) \notin V^{kf})$ **then**
        $V^{kf} \leftarrow V^{k} \cup \{o(k)\}$
      end if
      $E^{kf} \leftarrow E^{kf} \cup \{(o(k), i)\}$
    end for
    for all $((i_e, n) \in A^{kf}_{pin} | n = (d(k), t))$ **do**
      if $(d(k) \notin V^{kf})$ **then**
        $V^{kf} \leftarrow V^{kf} \cup \{d(k)\}$
      end if
      $E^{kf} \leftarrow E^{kf} \cup \{(i, d(k)\}$
    end for
  end for
end for

variable θ_p be 1 if route p is selected in the solution, and $\theta_p = 0$ otherwise. The path-based formulation for the MDVTSP is as follows:

$$\min \sum_{f \in F} \sum_{k \in K} \sum_{p \in \Omega^{kf}} c_p \theta_p \tag{4.1}$$

s.t.

$$\sum_{f \in F} \sum_{k \in K} \sum_{p \in \Omega^{kf}} a_{ip} \theta_p = 1, \qquad \forall i \in V^{kf} \tag{4.2}$$

$$\sum_{f \in F} \sum_{k \in K} \sum_{p \in \Omega^{kf}} q_f a_{ip} \theta_p \geq d_i \qquad \forall i \in V^{kf} \tag{4.3}$$

$$\sum_{p \in \Omega^{kf}} \theta_p \leq v_{kf}, \qquad \forall k \in K, \forall f \in F \tag{4.4}$$

$$\theta_p \in \{0, 1\}, \qquad \forall p \in \Omega^{kf} \tag{4.5}$$

Objective function (5.1) aims to minimize the total operational costs. Constraints (5.2) ensure that each trip is performed by exactly one vehicle. Constraint (5.3) guarantee that the peak demand of each trip is satisfied, while constraints (5.4) assure that the depot capacity for each vehicle type is respected.

The linear relaxation of model (4.1) – (4.6) is solved through column generation by repeatedly solving (i) a restricted master problem (RMP) with a subset of columns and (ii) a pricing subproblem to produce columns with negative reduced cost. Even if the linear relaxation of the model is solved to optimality by CG, it is not guaranteed that the resulting optimal solution is integral. We included one step for rounding the solution (the algorithm is better explained at Section 5.5.3).

4.1.2 Pricing Problem

Let dual variables π and σ correspond to constraints (4.3) and (4.4), respectively. Based on these dual variables, we have the following pricing problem:

$$\min \ -\sigma_{kf} + \sum_{(i,j)\in E^{kf}} (c_{ij}^{f} - \pi_j) x_{ij}^{kf} \tag{4.6}$$

s.t.

$$\sum_{k\in K} \sum_{f\in F} \sum_{j\in(i,j)\in E^{kf}} q_f x_{ij}^{kf} \geq d_i \qquad \forall i \in V^{kf} \tag{4.7}$$

$$\sum_{j:(j,i)\in E^{kf}} x_{ji}^{kf} - \sum_{j:(i,j)\in E^{k}} x_{ij}^{kf} = 0 \qquad \forall i \in (V^{kf} - \{o(k), d(k)\}), \forall k \in K, f \in F \tag{4.8}$$

$$x_{ij}^{kf} \in \{0,1\} \qquad \forall (i,j) \in E^{kf}, k \in K, f \in F \tag{4.9}$$

The pricing problem consists of finding the shortest path from origin depot $o(k)$ to same destination depot $d(k)$, where the peak demand of each trip cannot extrapolate the capacity of vehicle $f \in F$, characterizing the pricing problem as a resource-constrained shortest path problem (RCSP) due to constraints (4.7). The RCSP is a well-known NP-hard problem (MEHLHORN; ZIEGELMANN, 2000; BEASLEY; CHRISTOFIDES, 1989). Therefore, the pricing subproblem is also NP-hard, compromising the efficiency of the decomposition. A possible solution is also relax constraints (4.2). Let dual variable γ correspond to constraints (4.2). The new pricing problem can be formulated as follows:

$$\min \ -\sigma_{kf} + \sum_{(i,j)\in A^{kf}} (c_{ij}^{f} - \pi_j - q_f\gamma_j) x_{ij}^{kf} \tag{4.10}$$

s.t.

(5.7) – (5.8)

The resulting pricing consists of the shortest path problem from origin depot $o(k)$ to same destination depot $d(k)$ for vehicle type $f \in F$, an easy problem to be solved.

4.1.3 Modified Truncated Column Generation Algorithm

The developed truncated CG with variable fixing was based in the algorithm developed by (PEPIN et al., 2009). This algorithm requires three predefined parameters, namely Z_{min}, I, and Ω_{min}. The algorithm terminates early if the optimal value of the RMP has not decreased by more than Z_{min} in the last I iterations, in an attempt to avoid the well-known tailing off effect (LÜBBECKE; DESROSIERS, 2006). Parameter Ω_{min} is a threshold value in the rounding of variables θ_p. However, we introduced two changes

Algorithm 3 Modified Truncated Column Generation

Step 1 **(Initialization)**: Set parameters Z_{min}, I_{min}, $Num\_Stabilized$ and θ_{min}. Set $n \leftarrow 1, n_0 \leftarrow 1$ and $stabilized \leftarrow 0$. Add artificial variables δ^i in constraints (4.1) and (4.3) and with coefficients *big-M*.

Step 2 **(Restricted Master Problem):** Solve the RPM, obtaining a primal solution $(\theta_{\mathbf{n}}, \delta_{\mathbf{n}})$, and cost Z_n^{RMP}.

Step 3 **(Stabilization Test):** If $(stabilized > Num\_Stabilized)$ then go to step 6; otherwise go to step 4.

Step 4 **(Early Termination Test):** If $(n - n_0 > I_{min})$ and $(Z_{n_0}^{RMP} - Z_n^{RMP} < Z_{min})$ then go to step 6; otherwise go to step 5.

Step 5 **(Pricing Solution):** For each $k \in K$ and for each $f \in F$ do

Step 5.1 Update the costs of arcs A^{kf} using dual variables $(\pi_n, \sigma_n, \gamma_n)$. Solve a shortest path in graph G^{kf} using the SLF algorithm (BERTSEKAS, 1993).

Step 5.2 If the subproblem presents a negative reduced cost then set $\Omega_{n+1}^{kf} \leftarrow \Omega_n^{kf} \cup S_n^{kf}$, where S_n^{kf} is the solution of the subproblem, $n \leftarrow n + 1$. Solve the RPM with the updated set of columns. If $Z_{n-1}^{RMP} - Z_n^{RMP} = 0$ then $stabilized \leftarrow stabilized + 1$ else $stabilized \leftarrow 0$. Go to step 3.

Step 6 **(Master Problem Feasibility Test):** If $(\delta_{\mathbf{n}} \neq 0)$ STOP; else go to step 7.

Step 7 **(Integrality):** If $(\theta_{\mathbf{n}} \in \{0, 1\})$ then the solution was found. Return θ_n. Stop.

Step 8 **(Variable Fixing):** Perform a rounding of the no-integer variables in $\theta_{\mathbf{n}}$. If $(\theta_{p,n} \geq \theta_{min})$ set $\theta_p \leftarrow 1$. If no such variables exist set the highest value in $\theta_{\mathbf{n}}$ as one. Set $n \leftarrow n + 1$ and $n_0 \leftarrow n$. Go to Step 2.

to this algorithm based on initial experimentation, customizing it to the MDVTSP. We introduced a fourth parameter, $Num\_Stabilized < I$. This parameter is used to stop generating columns when the objective function of the RPM remains unaltered in the last $Num\_Stabilized$ iterations and perform rounding to perturb the solution. This change was necessary, since an acceptable value of I was much higher (around 30) than the obtained by Pepin et al. (2009) (around 5) for solving the MDVSP. The second change was in the way the columns are inserted in the RMP. The algorithm described by (PEPIN et al., 2009) inserts at most $|K|$ columns at each iteration of the CG procedure, depending on the sign of the reduced cost of each $k \in K$ subproblem. The master problem, with these new columns, is solved once at the beginning of the next iteration of the CG procedure. Our approach uses the traditional way of selecting columns to insert in the current master problem. However, it solves the master problem right after a new column is inserted. The main rationale behind this idea is to select better columns to insert in the current master problem each time a subproblem is solved, always using updated duals. Although this procedures solves initially a higher number of master problems, the use of updated duals accelerates the convergence of the CG procedure, avoiding the generation of many similar columns.

The overall modified truncated CG with variable fixing approach is outlined as follows.

Step 1 **(Vehicle Scheduling Network):** Generate the time-space network.

Step 2 (TSN Reduction): Reduce the TSN by applying the reductions suggested by Kliewer, Mellouli e Suhl (2006).

Step 3 (Network Concersion): Convert the time space network to a connection network, using Algorithm 2.

Step 4 (Column Generation): Solve the problem using Algorithm 7

4.2 Accelerating Heuristic for Large-Scale Instances

The branch-and-price algorithm has been successful in obtaining good quality solutions in many combinatorial optimization problems (BARNHART et al., 1998). However, this approach still presents some difficulties in solving large instances. Moreover, CG has often presented some difficulties in obtaining solutions with efficiency, even for medium size instances, due to its well-know stabilization problems (OUKIL et al., 2007). A CG-based heuristic is then developed in an attempt to increase simultaneously both shortcomings of the branch and price framework. First we propose a state space reduction procedure to decrease the required number of variable to be solved by the column generation algorithm. Next, we propose a routine to offer better initial solutions than the traditional use of big-M.

In vehicle scheduling problems, it is natural that several trips are very expensive to be carried out, since trips that are close geographically can be far apart due to times in which they must start, while others may be compatible in terms of timing, but geographically infeasible. Several variables in the problem become too costly to be considered as viable alternatives in an optimal or close to optimal solution, both due to distance or timing issues. As a consequence, although the complete solution space is very large, the set of variables with high chance of being in the final solution is much smaller. In fact, a large number of variables has small chances of being considered as good options to be included in the final solution. Based on this observation, we developed a procedure that reduce the complete state space to a simpler, more manageable set, focusing attention on a selected portion of the complete state space.

The goal of this procedure is to identify "relevant" variables that are selected by any solution of all $|F| \times |K|$ SDVSPs. The reasoning behind this procedure is quite intuitive. If a variable is not chosen as a solution considering $|F| \times |K|$ SDVSPs, then it would have small chance to be chosen as a candidate solution when considering the whole problem. This procedure consists in efficiently solving $|F| \times |K|$ individual SDVSPs. The arcs found in any of the solved SDVSPs configures a reduced connection graph, represented by matrix $\mathbf{MDVTSP}_r^{kf}$, with a similar structure to matrix $MDVTSP^{kf}$, in which $MDVTSP_r^{kf}[i][j] = MDVTSP^{kf}[i][j]$ if arc (i, j) is in the solution of any

$SDVSP^{kf}, k \in K, f \in F$; and $MDVTSP_r^{kf}[i][j] = -1$, otherwise. It should be noticed that pull-in and pull-out arcs cannot be reduced by introducing further severe instabilities in the CG solution process. Each SDVSP problem is structured by matrix $\mathbf{VSP}^{fk}$, easily obtained from $\mathbf{MDVTSP}^{kf}$ by taking into consideration only the pull-in and pull-out trips to and from depot k, respectively, and vehicle type f. Each SDVSP can be modeled as an assignment problem (AP) following Paixão e Branco (1987). There are several algorithms specially designed to solve linear AP. Based on results reported by Dell'Amico e Toth (2000), the algorithm LAPJV (JONKER; VOLGENANT, 1987) was employed to solve each SDVSP. The selection procedure can be outlined as follows.

Algorithm 4 State space reduction procedure

$MDVTSP_r^{kf}[i][j] \leftarrow MDVTSP^{kf}[i][j], i = 1, \ldots, |K|, j = 1, \ldots, |N| + |K|$
$MDVTSP_r^{kf}[i][j] \leftarrow MDVTSP^{kf}[i][j], i = |K| + 1, \ldots, |N| + |K|, j = 1, \ldots, |K|$
for all $k \in K$ **do**
  for all $f \in F$ **do**
    Generate matrix $\mathbf{VSP}^{kf}$ from $\mathbf{MDVTSP}^{kf}$
    $Solution \leftarrow$ LAPJV($VSPkf$)
    if arc $(i, j) \in Solution$ **then**
      $MDVTSP_r^{kf}[i][j] \leftarrow MDVTSP^{kf}[i][j]$
    end if
  end for
end for

The LAPJV algorithm has a complexity of $O(n^2 . \log n)$. As algorithm 4 is executed $|F| \times |K|$ times, the complexity of the selection procedure is $O(|F| \times |K| n^2 . \log n)$, where $n = 2|K|$.

Column generation requires an initial solution. Initially, we replicated the use of artificial variable penalized by a big-M cost (see Algorithm 7) as defined in Pepin et al. (2009). Although a good initial solution cannot guarantee a good convergence process (LÜBBECKE; DESROSIERS, 2006), we decided to use the paths obtained by Algorithm 4 as an initial set of columns to the RMP. Its primal optimal solution is used to compute an initial solution for the MDVTSP, offering an upper bound on the integer optimal value. Its dual solution provides a lower bound on the RMP.

The overall CG-based heuristic framework for solving the MDVTSP is outlined as follows. Steps 1, 2, and 3 are the same as the corresponding steps in the modified truncated column generation with variable fixing approach.

Step 4 (State Space Reduction): Generate matrices $\mathbf{MDVSP}_r^{kf}$ by applying Algorithm 4.

Step 5 (Initial Solution): Save the solutions obtained in Step 4 in set S^0.

Step 6 (Column Generation): Apply Algorithm 7 in the reduced matrices $\mathbf{MDVSP}_r^{kf}$, using S^0 as the initial solution.

Our preliminary experiments show that both generating an initial solution and reducing the solution state space approach may be helpful for speeding up the CG in vehicle scheduling problems.

5 Computational Experiments

The main objective of the computational experiments was to compare the performance of the developed solution approaches, in terms of quality and efficiency of the solution, using random generated instances.

The algorithms were implemented in C++. All experiments were carried out in an Intel Xeon CPU E5-1603, 2.80 GHz, 16 GB de RAM. We used CPLEX 12.5 to solve all mixed linear programming models. The following notation is used to indicate the implemented algorithms: (i) MTCG: Modified truncated CG with variable fixing as described in Section 5.5.3; (ii) IS: Same as MTCG using the initial solutions obtained using Algorithm 4, but without state space reduction; (iii) SSR: Same as MTCG employing the state space reduction procedure; and (iv) IS+SSR: The complete framework developed in Section 5.3, using the state space reduction procedure and initial solutions for the CG.

5.1 Experiments Configuration

Although Carpaneto et al. (1989)'s random instance generation method has been widely used in the MDVSP literature (RIBEIRO; SOUMIS, 1994; PEPIN et al., 2009), there is no method for the generation of MDTVSP instances, considering both TSN and demand. We had to develop our own method. Let $\rho_1, \rho_2, \ldots, \rho_{20}$ be the station nodes (i.e. points where trips can start and finish) of a transit network. The stations were generated as the nodes of a planar graph employing Dorogovtsev e Mendes (2002)'s algorithm. The algorithm starts by creating a triangle, using three nodes and tree edges, and then add one node at a time. Each time a node is added, an edge is chosen randomly and the node is connected via two new edges to the two extremities of the chosen edge. The process generates a power-low degree distribution, as nodes that have more edges have more chances to be selected. The corresponding distance between relief points ρ_a and ρ_b, $d(\rho_a, \rho_b)$, were computed using Floyd-Warshal's all-pair shortest path lengths algorithm. To simulate the trips, we generated for each trip $i, i = 1, \ldots, n$ the starting and relief points randomly selected from the station nodes. The starting time st_i is a uniformly random integer in interval (0,23). However, in order to represent a realistic frequency of trips that takes into account the peak hours, a random process was included to define if a trip will be in the timetabling. The frequency of trips is based on a curve obtained by a weighted sum of three Gaussian distributions, with averages of 7, 12, and 18, and standard deviations of 2, 1.5, and 3, respectively. These values were obtained based exiting timetables of several cities in Brazil. Averages represent the peak hours. The weights of each distribution are 4.5, 1, and 5, respectively, guaranteeing that the resulting curve,

called as frequency trip (FT) function, has a total area equal to 1. Figure 2 exemplifies FT. An additional uniformly integer in interval (0,1), u, is generated along with st_i. Trip i is included in the timetabling if $u \geq FT(st_i)$. Since the ending time et_i of trip i must include a travel time between stations ρ_a and ρ_b, and a dwell time at vehicle stops, we generated $et_i = (d(\rho_a, \rho_a) + Dist_{min}) * rnd(0, 150) * 60$, where $Dist_{min} = 180$s, and rnd is a function that generates uniformly random integer in interval (a, b). The demand of trip $i, D(i)$, was computed using $D(i) = (rnd(0, Q) * FT(st_i)) + D_{min}$, where Q is the maximum vehicle type capacity, and D_{min} is a minimum demand value. In our experiments, $D_{min} = 20$. Table 4 presents the fixed costs and capacity of each vehicle type used in the experiments based on information provided by the transit network company of São Paulo, Brazil.

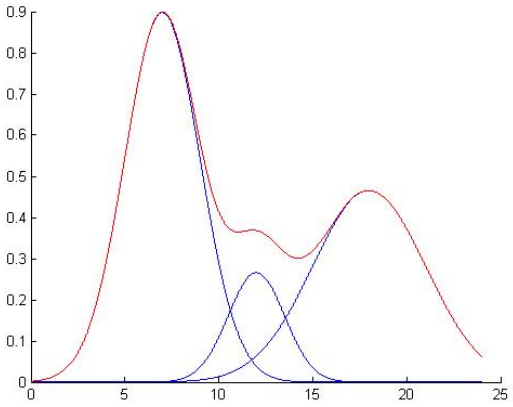

Figure 2 – Frequency trip function

Vehicle Type	Fixed Cost, FC (in \$)	Capacity
Microbus	148,517.54	21
Minibus	185,000.00	40
Classic	262,000.00	53
Standard 13	310,000.00	83
Standard 15	582,850.47	110
Articulated bus	628,844.20	120
Biarticulated I	799,164.00	140
Biarticulated II	921,518.00	190

Table 4 – Specification of each vehicle type

Travel costs c_{ij}^f were defined based on the vehicle type, and included the travel and waiting times. The vehicle type was considered using cost factors directly related with the vehicle fixed cost and the number of types considered. Since the experiments involved 3, 5, and 8 vehicle types, the following cost factor vectors ($\alpha_{f|F|}$) were computed: [0.8, 1, 1.2], [0.4,0.7,0.8, 1, 1.2], and [0.4, 0.5, 0.6, 0.7, 0.8, 1, 1.2, 1.5], respectively. The travel costs were computed as follows:

- $c_{ij}^f = \alpha_{f|F|} * (st_j - et_i) * 8$, for the deadheading trips between two compatible trips i and j.

- $c_{ij}^f = \alpha_{f|F|} * (st_j - et_i) * 4$, for the waiting trips between two compatible trips i and j.
- $c_{kj}^f = \alpha_{f|F|} * (5,000 + d(o(k), \rho_j))$, for the pull-out trips from depot $k \in K$.
- $c_{jk}^f = \alpha_{f|F|} * (5,000 + d(\rho_i, d(k))$, for the pull-in trips to depot $k \in K$.

We have set the fixed costs of the depots pull-in and pull-out arcs high enough (a total of 10,000) to allow the minimum cost flow problem determine the optimal number of vehicles, following Pepin et al. (2009).

The settings of CG parameters, namely Z_{min}, I, and Ω_{min}, were carried out by experimentation, using randomly generated instances consisting of 500 to 5000 trips, and up to 128 depots. Different combinations of these three parameters (Z_{min}=0,100,10000; I=2,5,10,50; and Ω_{min} in interval [0.7,0.9]) led to different solutions and CPU times. The best quality solutions were obtained with small values of Z_{min} and large values of I and Ω_{min}, but, in general, with low efficiency. The best solutions were found with $Z_{min} = 0$, in which step 4 of algorithm 7 is not fired and the solution process is not terminated prematurely. Concerning the number of minimum iterations, to increase I has, in general, improved the quality of the solution. However, the solution efficiency was negatively affected by the increase of I. Good compromised solutions, in terms of quality and CPU times, were obtained with $I = 5$, Regarding Ω_{min}, our experiments were not conclusive in terms of finding a clear correlation between this parameter and the solution quality. The behavior of this parameter was instance dependent. Overall good solutions, in terms of quality and efficiency, were obtained with this parameter set to 0.8.

For the comparison of the algorithms, random instances were generated with $ST = \{500, 1000, 1500, 2000, 3000\}$, $K = \{4, 8, 16, 32\}$, and $F = \{3, 5, 8\}$. For each combination of trips, depots, and vehicle types, five instances were generated, totaling 300 instances. However, some of the instances with 3000 trips, 32 depots, and 8 vehicles were not solved by CPLEX due to the excessive number of variables and constraints. For these instances, results are not reported.

5.2 Results

Table 5 compares the performance of algorithms MTCG, IS+SPR, SPR, and IS for five instances of each problem. The first three columns display the number of trips, depots, and vehicle types, respectively. The next three columns show the average gap (in %) from the best solutions obtained by MTCG, computed as follows $GAP = 100 \times \frac{S_A}{S_{MTCG}}$, where S_{MTCG} is the average best solutions obtained by algorithm MTCG, and S_A is the average best solution obtained by the algorithm being compared, say algorithm A. The last four columns show the average CPU times in seconds, excluding input and output

time, for each algorithm. The tables show that algorithms IS+SPR, SPR, and IS, in general, offer very good solution quality (some with negative gaps) very efficiently. For some instances, the high number of possible routes and the use of artificial variables in the MTCG solution process led to a slow convergence of the algorithm. The former issue also caused an excessive rounding of the decision variables, setting a high number of θ_p to 1. As a consequence, constraints (4.3) make difficult the creation of new routes, stagnating the solution around a not so good value. Based on the results reported in this table, it is possible to state that the use of initial solution for the CG algorithm had a significant effect in the efficiency of the algorithms, while the state space reduction had a major effect in the quality of the solution.

Analyzing the solution quality, it is important to mention that all algorithms found the minimum number of vehicles for all instances. Table 5 shows that algorithms IS+SSR, SSR, and IS found better average solutions than MTCG for several instances. Algorithms IS+SSR, SSR, and IS improved, on average, the best solution in comparison to MTCG in around 50%, 40%, and 71%, respectively, of the tested instances. The algorithms improved quite significantly the solution quality of some instances, obtaining minimum average gaps of 9.63%, 6.54%, and 8.33%, respectively; while the maximum average gaps were 2.20%, 1.75%, and 5.35%, respectively. The improvements in the solution quality happened as the number of trips and vehicles types increased. Worse solutions were obtained as the number of depots increases. Algorithms IS+SSR and SSR obtained worse solutions for all instances with 32 depots.

The average CPU time for all algorithms is highly dependent on the instance size, being the number of trips the most influential parameter. Table 5 shows that either the use of initial solutions for the truncated CG algorithm or the state space reduction led to efficient algorithms. The use of these procedures decreased, in several instances, the number of required iterations to find a good solution. Algorithm IS+SSR was the most efficient algorithm for all instances, but one (500 trips, 32 depots, and 8 vehicle types). If we compare with MTCG, algorithms IS+SSR, SSR, and IS reduced the average CPU times in 63.2%, 18.6%, and 59%, on average. Table 6 presents the CPU time speedup factors for algorithms IS+SSR, SSR, and IS, taking MTCG as the comparison basis. We can see that when the number of trips becomes larger, the heuristics provided significant CPU time reductions. However, CPU usage reduction decreased with the number of depots and vehicles. From this table, it is possible to conclude that the benefits of the preprocessing procedures is basically asymptotically dependent on the number of depots. The best performance of algorithms IS+SSR and IS were obtained for instances with 3000 trips, 4 depots, and 3 vehicle types. The worse performance occurred in instances with 500 trips, 32 depots, and 4 vehicle types. Algorithm SSR did not obtain relevant gains in the CPU speedup factor in comparison with IS+SSR and IS, being the less affected algorithm by the

problem dimensions. Algorithms IS+SSR and IS increased significantly the convergence process of the CG for up to 16 depots.

Table 7 shows that the CPU times obtained for solving the MDVTSP are quite competitive with values reported to solve its counterpart MDVSP (OTSUKI; AIHARA, 2014; PEPIN et al., 2009), considering the same number of trips and depots. The competitive CPU times were justified by the combination of TSN, state space reduction, truncated CG, and special procedures for generating good initial solutions for the CG algorithm. The CPU times decreased with the increase in the use of these approaches. In summary, we can conclude that the developed algorithms are computationally efficient in solving the MDVTSP.

Overall, our developed implementations were quite efficient. Algorithm IS+SSR obtained the average best solutions of all tested algorithms, considering both quality and efficiency. IS+SSR obtained better quality solutions with a speedup factor of 4.4, on average, in comparison with algorithm MTCG, a more traditional CG implementation for the MDVTSP. The simultaneous use of good initial solutions and state space reduction significantly increased the convergence process of the CG method, decreasing or eliminating the well-known tailing effect (LÜBBECKE; DESROSIERS, 2006). However, the final selection among these three variants is case dependent, and it will depend on further experimentation, with a special attention on the setting of CG parameters.

5.3 Tests on a Real-World Instance

We used data from a public transit company located in the city of Santa Maria, in the south of Brazil, to evaluate our framework in a real-world instance. The transit system is run by ATU, a consortium formed by five public transportation companies, namely Viação Centro Oeste, Expresso Nossa Senhora das Dores, Expresso Medianeira, Gabardo Transporte, and Salgado Filho Transporte. The data set provided by ATU are from the area “Federal University of Santa Maria", comprising 20 different lines, 9 stations, and 529 daily trips, transporting on an average class day around 26.824 passengers, 25% of the total number of passengers transported daily in Santa Maria. This area was selected since the average peak demand of each trip was previously defined. This information is quite difficult to obtain, since transport companies only register the paying passengers for tariff definition. However, several passengers are exempt from charges due to Brazilian legislation, such as elder people, police, postman, etc. Three of the analyzed lines area are the busiest in the city. The area contains five depots, the garages of the companies in the consortium. The involved distances, between the depots and stations, and among the stations, were provided by ATU. The fleet is composed by three different vehicle types as follows: type A, corresponding to an articulated bus, with capacity for 140 passengers;

Table 5 – Comparison results

			Solution Gaps			CPU Time (s)			
Trips	Depots	Vehicles	IS+SSR	SSR	IS	MTCG	IS+SSR	SSR	IS
500	4	3	-2.74%	-0.61%	2.49%	16.4	2.4	12.6	1.4
500	8	3	0.62%	0.88%	2.44%	21.2	3.8	15.4	4.2
500	16	3	0.30%	0.32%	-0.02%	21.8	9.2	20.2	10.4
500	32	3	0.40%	0.41%	-0.02%	25.0	20.4	26.0	21.8
1000	4	3	-2.43%	-1.90%	3.37%	87.4	7.2	56.4	7.8
1000	8	3	-1.68%	-1.18%	-	99.4	21.2	68.0	26.4
1000	16	3	0.28%	0.30%	-0.04%	98.2	43.4	90.2	48.2
1000	32	3	0.27%	0.27%	-0.01%	134.4	80.0	130.4	97.2
1500	4	3	2.20%	-1.58%	5.35%	332.6	26.4	179.8	26.6
1500	8	3	-0.60%	0.68%	2.64%	353.6	53.4	219.8	53.0
1500	16	3	0.45%	0.48%	-0.02%	359.8	119.6	278.8	139.0
1500	32	3	0.61%	0.61%	-0.01%	621.8	242.2	453.0	328.0
2000	4	3	1.37%	-1.33%	4.71%	633.0	61.4	347.2	60.8
2000	8	3	-4.78%	-2.76%	2.03%	693.2	114.6	445.4	102.4
2000	16	3	0.22%	0.24%	-0.13%	716.0	241.8	593.2	301.8
2000	32	3	0.31%	0.32%	-	871.4	491.2	887.6	581.6
3000	4	3	1.57%	-1.56%	3.91%	2419.0	140.4	993.6	141.8
3000	8	3	-6.65%	-3.82%	-0.27%	2448.6	331.6	1243.8	288.8
3000	16	3	-1.31%	-1.28%	-1.79%	2550.2	715.6	1670.6	773.4
3000	32	3	0.40%	0.40%	-0.01%	3436.6	1419.0	2494.6	1707.2
500	4	5	-4.98%	1.24%	1.51%	18.8	2.8	15.4	3.0
500	8	5	-1.59%	0.27%	-1.98%	26.0	8.0	20.4	9.8
500	16	5	0.46%	0.51%	-0.01%	27.8	15.4	33.4	16.2
500	32	5	0.53%	0.52%	-0.01%	37.4	29.4	43.4	38.2
1000	4	5	-3.48%	-0.34%	3.09%	107.6	16.0	69.4	13.8
1000	8	5	-4.67%	-0.83%	-3.64%	117.6	30.6	89.6	39.6
1000	16	5	0.26%	0.35%	-0.01%	127.8	55.0	133.8	62.2
1000	32	5	0.30%	0.30%	-	209.8	114.8	213.8	136.6
1500	4	5	-6.16%	-2.90%	-1.01%	353.2	33.8	215.0	37.8
1500	8	5	-2.80%	1.75%	-1.07%	420.2	95.8	252.6	95.6
1500	16	5	0.44%	0.46%	0.14%	424.2	196.8	365.0	210.8
1500	32	5	0.61%	0.62%	-0.01%	711.2	362.4	632.8	438.2
2000	4	5	-7.15%	-3.32%	-0.93%	704.6	80.8	433.6	77.2
2000	8	5	-3.30%	0.31%	-2.29%	765.0	161.2	540.2	178.2
2000	16	5	-0.21%	-0.47%	-0.70%	909.8	345.0	688.0	432.2
2000	32	5	0.32%	0.33%	-	978.0	643.6	1184.2	741.6
3000	4	5	-4.11%	-1.62%	0.08%	2707.0	199.8	1174.2	211.0
3000	8	5	-8.64%	-6.54%	-5.22%	2346.8	417.6	1477.6	418.8
3000	16	5	0.58%	0.52%	0.32%	2932.4	1034.0	2015.2	1174.8
3000	32	5	-4.78%	-4.78%	-5.17%	3522.8	1997.2	3283.0	2055.8
500	4	8	-6.05%	-1.15%	-4.60%	19.6	5.2	19.4	4.8
500	8	8	-1.84%	-0.80%	-1.83%	32.4	10.4	28.8	14.0
500	16	8	0.13%	0.25%	-0.16%	35.8	22.0	36.2	28.6
500	32	8	0.67%	0.67%	0.01%	60.6	65.4	62.8	67.0
1000	4	8	-8.78%	0.33%	-5.17%	112.0	22.6	84.0	24.0
1000	8	8	-1.70%	-0.19%	-1.80%	144.4	58.2	119.4	69.4
1000	16	8	0.30%	0.67%	0.15%	168.2	90.6	192.8	102.8
1000	32	8	0.49%	0.51%	-0.01%	240.8	210.4	276.2	222.4
1500	4	8	-6.59%	1.07%	-0.54%	384.6	62.6	243.2	50.2
1500	8	8	-1.63%	1.26%	-1.64%	443.4	155.6	324.4	143.2
1500	16	8	0.68%	0.79%	-0.01%	503.8	252.0	544.0	317.4
1500	32	8	0.63%	0.63%	-	822.0	509.8	857.2	669.6
2000	4	8	-4.65%	-0.87%	-1.46%	742.0	117.4	502.0	104.0
2000	8	8	-6.28%	-3.53%	-6.09%	847.0	305.4	661.4	243.6
2000	16	8	-0.04%	0.14%	-0.52%	1006.8	593.2	1033.0	600.2
2000	32	8	0.36%	0.37%	-	1342.6	1121.2	1620.4	1198.8
3000	4	8	-9.63%	-2.37%	-2.16%	2672.2	332.0	1395.6	289.6
3000	8	8	-8.33%	-5.04%	-8.33%	2713.6	817.8	1791.0	756.2
3000	16	8	0.61%	1.20%	0.35%	3271.2	1707.4	2706.0	1686.2

Table 6 – Speedup factors

Trips	Depots	Vehicles	IS+SSR	SSR	IS
500	4	3	6.83	1.30	11.71
500	8	3	5.58	1.38	5.05
500	16	3	2.37	1.08	2.10
500	32	3	1.23	0.96	1.15
1000	4	3	12.14	1.55	11.21
1000	8	3	4.69	1.46	3.77
1000	16	3	2.26	1.09	2.04
1000	32	3	1.68	1.03	1.38
1500	4	3	12.60	1.85	12.50
1500	8	3	6.62	1.61	6.67
1500	16	3	3.01	1.29	2.59
1500	32	3	2.57	1.37	1.90
2000	4	3	10.31	1.82	10.41
2000	8	3	6.05	1.56	6.77
2000	16	3	2.96	1.21	2.37
2000	32	3	1.77	0.98	1.50
3000	4	3	17.23	2.43	17.06
3000	8	3	7.38	1.97	8.48
3000	16	3	3.56	1.53	3.30
3000	32	3	2.42	1.38	2.01
500	4	5	6.71	1.22	6.27
500	8	5	3.25	1.27	2.65
500	16	5	1.81	0.83	1.72
500	32	5	1.27	0.86	0.98
1000	4	5	6.73	1.55	7.80
1000	8	5	3.84	1.31	2.97
1000	16	5	2.32	0.96	2.05
1000	32	5	1.83	0.98	1.54
1500	4	5	10.45	1.64	9.34
1500	8	5	4.39	1.66	4.40
1500	16	5	2.16	1.16	2.01
1500	32	5	1.96	1.12	1.62
2000	4	5	8.72	1.63	9.13
2000	8	5	4.75	1.42	4.29
2000	16	5	2.64	1.32	2.11
2000	32	5	1.52	0.83	1.32
3000	4	5	13.55	2.31	12.83
3000	8	5	5.62	1.59	5.60
3000	16	5	2.84	1.46	2.50
3000	32	5	1.76	1.07	1.71
500	4	8	3.77	1.01	4.08
500	8	8	3.12	1.13	2.31
500	16	8	1.63	0.99	1.25
500	32	8	0.93	0.96	0.90
1000	4	8	4.96	1.33	4.67
1000	8	8	2.48	1.21	2.08
1000	16	8	1.86	0.87	1.64
1000	32	8	1.14	0.87	1.08
1500	4	8	6.14	1.58	7.66
1500	8	8	2.85	1.37	3.10
1500	16	8	2.00	0.93	1.59
1500	32	8	1.61	0.96	1.23
2000	4	8	6.32	1.48	7.13
2000	8	8	2.77	1.28	3.48
2000	16	8	1.70	0.97	1.68
2000	32	8	1.20	0.83	1.12
3000	4	8	8.05	1.91	9.23
3000	8	8	3.32	1.52	3.59
3000	16	8	1.92	1.21	1.94

Table 7 – Comparison of the proposed methods with other methods

Trips	Depots	Average CPU Times (s)					
		Pepin et al. (2009)	Otsuki e Aihara (2014)	MTCG	IS+SSR	SSR	IS
500	4	77	74	18.3	3.5	15.8	3.1
1000	4	612	612	102.3	15.3	69.9	15.2
1500	4	2012	2012	356.8	40.9	212.7	38.2
500	8	119	109	26.5	7.4	21.5	9.3
1000	8	857	787	120.5	36.7	92.3	45.1
1500	8	3085	2800	405.7	101.6	265.6	97.3

type B, the most commonly used, with capacity for 95 passengers; and type C, a smaller vehicle able to transport 84 passengers. Vehicle A is the most expensive bus to operate, both in fixed and operational costs, being vehicle C the least expensive. Unfortunately, several costs required by the framework needed to be estimated. ATU does not compute several involved model parameters, such as waiting costs in stations and depots, and daily fixed costs. Our estimations were based on information provided by ATU, but they are not accurate.

Table 8 presents the results obtained by the four developed solution methods for the real instance on an average class day. The manual method refers to the current scheduling adopted by ATU. All methods used the same set of parameter values. It should be noticed that the values presented in column Solution are estimated ones, due to the inaccuracy of some parameters. As a consequence, only relative comparisons are appropriate concerning the total costs obtained by each method. Based on the CPU times presented in Table 8, it is possible to conclude that this instance is much more difficult to be solved than the correspondent (with similar number of trips, depots, and vehicles) random generated ones.

Table 8 – Computational results for the real instance

Method	Solution	Time	# Vehicles			
		(sec.)	A	B	C	Total
Manual	1385484	—	5	45	4	58
MTCG	1043710	1429	6	10	33	49
IS+SSR	945538	26	5	11	31	47
SSR	876131	76	4	9	29	42
IS	1087780	22	10	17	19	46

Comparing only the heuristic methods, the best quality solution was obtained by SSR, both in costs and in the number of vehicles. IS was the most efficient method, but with the worst solution quality. MTCG obtained the second worst solution among the heuristic approach, requiring excessive average CPU times in comparison with the other optimization approaches. MTCG obtained a solution with excessive number of vehicles, increasing the fixed costs. It seems that the MTG convergence problems, related in the previous section, were aggravated in this real-world instance. IS+SSR obtained the best

compromise solution, taking into consideration quality and efficiency. However, considering that the MDVTSP is a typical planning problem, and the high efficiency of all heuristics that employ an accelerating preprocessing routine (less than 1.5 minutes), we can consider SSR as the best heuristic method for this instance. As a consequence, only SSR will be used for further analysis.

All heuristic methods have outperformed the manual planning in terms of solution quality. Total costs were reduced in around 31.75%, considering SSR as the best solution. The heuristic approach has reduced the number of vehicles from 58 to 42 vehicles to fulfill the 529 trips in the timetabling. The capability of the heuristic methods to assign several trips to a single bus, from different lines, is a good explanation for this significant reduction in the number of vehicles. With manual planning, a bus is usually designated to only one line. Several of them stay idle during the day, waiting in the depot to leave in rush hours (6:30–9:00, and 17:30-20:00). This practice in manual planning also leads to longer distances traveled on deadheading trips, since a bus needs to travel back to the depot, resulting also in unnecessary additional deadheading trips. The optimization approaches are capable of assigning a bus from the starting point of the next compatible trip, independently of lines, reducing the distances in deadheading trips. Moreover, as the heuristic methods take in consideration peak demands per trip, they obtained a much more economical distribution of vehicle types than the manual method. In order to avoid excess of passengers in a trip, ATU prioritizes the use of vehicle type B. As peak demands of trips out of the rush hours are lower than the capacity of vehicle C, the heuristic methods focused on this vehicle type, restricting the use of more expensive buses to rush hours. It is important to notice that the real savings might be smaller, since we are not considering the crew scheduling in the obtained solutions by the optimization approaches. This is note the case for the manual solution. The consortium is in the process of implementing some results from the optimization methods.

6 Conclusions

This paper presents several column based algorithms to solve the MDVTSP. To speed up the solution of very large instances, we developed a state space reduction procedure that selects, from the whole feasible solution space, a good set of arcs, based on the solution of $|F| \times |K|$ SDVSPs. The selected set of arcs is also used to generate initial columns in the posterior application of a modified truncated column generation algorithm. With the use of these preprocessing procedures, the MDVTSP can be solved with efficiency and efficacy in comparison with a more traditional CG implementation.

From computational experiments performed on randomly generated data, it is possible to observe that the algorithm that combines state space reduction and initial solution obtained, on average, the best solutions, requiring the smallest CPU times, overcoming all remaining algorithms for the set of analyzed instances. This implementation is more efficient than the state-of-the-art in MDVTSP solution, considering the same number of depots and trips. However, in a real case instance, the best solution was obtained by the heuristic that uses only the state space reduction (SSR). Comparing with the current manual scheduling used by the transport consortium, savings of around 31% and 35% were obtained in total costs and in the number of vehicles, respectively.

Future research is directed towards using the heuristic framework to develop novel approaches for solving both the real time vehicle recovery scheduling problem and the integrated crew and vehicle scheduling problem in transportation contexts where heterogeneous fleet is an relevant characteristics. These two problems are very complex ones, in which the MDVTSP needs to be solved several times as a subproblem.

Part III

A Novel Efficient Approach for the Real-Time Multi-Depot Vehicle Type Rescheduling Problem

Abstract

The multiple-depot vehicle type rescheduling problem (MDVTRSP) is a dynamic extension of the classic multiple-depot vehicle scheduling problem (MDVSP), where heterogeneous fleet is considered. The MDVTRSP consists of finding a new schedule given that a serious disruption occurred in a previous scheduled trip, simultaneously minimizing the transportation costs and the deviation from the original schedule. Although several mathematical formulations and solution methods have been developed for the MDVSP, the MDVTRSP is still unexplored. In this paper, we introduced a formulation for the problem oriented to public transportation, based on certain assumptions, and developed a heuristic solution method, employing time-space network, truncated column generation (TCG), and preprocessing procedures. The solution method has been implemented in several algorithm variants, combining different developed preprocessing procedures. Computational experiments on randomly generated instances were performed to evaluate the performance of the developed algorithms. The best solutions, considering simultaneously quality and efficiency, were obtained in the heuristics involving state space reduction.

Keywords: vehicle rescheduling, heterogeneous fleet, multi-depot, column generation.

1 Introduction

Public transportation systems are susceptible to disruptions, such as vehicle breakdowns and traffic accidents. For example, a regional bus transportation system in Arizona has 37 fixed routes; with a fleet of 189 buses, that serve over one thousand timetabled trips each weekday. In June 2005, the vehicles in the fleet traveled 622,198 miles. On the average, bus failures (e.g., engine problems, air conditioner malfunctions, traffic accidents and tire-wheel issues) were reported approximately every 5,000 miles resulting in over 100 repair calls for that month (Sun Tran, 2005). While minor vehicle failures can be repaired quickly, serious failures require long repair times and in general result in towing the disabled vehicle for lengthy repairs and long-term maintenance.

The focus of this paper is on the real-time schedule recovery for the case of serious vehicle failures. Such vehicle breakdowns require that the passengers from the disabled vehicle and those expected on the remaining part of the trip be picked up. In addition, since the disabled vehicle may have future trips assigned to it, the given schedule may be deteriorated to the extent where the fleet plan may need to be adjusted in real-time depending on the current state of what is certainly a dynamic system. Usually, without the help of a rescheduling algorithm, the dispatcher either cancels the trips that are initially scheduled to be covered by the disabled vehicle (when there are upcoming future trips planned that could soon serve the expected demand for the canceled trips), or simply dispatches an available vehicle from a depot. In both cases, there may be considerable delays introduced. This manual approach may result in a poor solution. The implementation of new technologies (e.g., automatic vehicle locaters, the global positioning system, geographical information systems, and wireless communication) in public transit systems makes it possible to implement real-time vehicle rescheduling algorithms at low cost.

In some sense, the vehicle rescheduling problem (VRSP) can be viewed as a dynamic version of the classic vehicle scheduling problem (VSP) where assignments are generated dynamically (LI; MIRCHANDANI; BORENSTEIN, 2007). Although the VSP is a well-studied problem, dynamic versions present peculiarities and diversity of situations that make the VRSP and its variant very difficult to model and solve. The rescheduling problem must be solved in short computational times, so that the disruption is controlled as soon as possible. Another issue on developing a new schedule is the associated crew scheduling problem. If the new schedule is very different from the current one, then it might be difficult to assign crews to trips with which they are familiar. Thus, having minimal changes to the current schedule should be a consideration in rescheduling. Finally,

note that a vehicle breakdown could lead to delays of multiple transit trips. The current trip that is directly affected by the breakdown is certainly delayed. Other trips may also be delayed by the vehicle breakdown; a typical instance is when the next trip that the disabled vehicle is scheduled to cover is far from the depot as well as from other currently operating vehicles on the road. Normally, the nominal time describing a scheduled trip includes some slack times to allow for small delays so that small variations in starting times can be easily tolerated. However, large delays are usually not acceptable by both potential passengers and transit system regulators. It is also worth mentioning that multiple vehicles can break down simultaneously, for example under poor weather conditions; the developed method should account for these extreme, but frequent, situations to be useful in real-world applications.

Although there are some developed modeling tools to deal with the real-time vehicle recovery problem (VISENTINI et al., 2014), the existing formulations do not recognize important aspects related with the problem. Some of them limit the disruptions only to delays, others apply algorithms and heuristics which performances are incompatible with the dimensions of real-world problems (involving thousands of trips and vehicles), while some neglects schedule disruption costs and impacts of excessive changes on the off-line scheduling. Under these approaches, the vehicle rescheduling is only applied to very limited real-world situations.

This paper proposes a formulation for the multi-depot vehicle type rescheduling problem (MDVTRSP) and a heuristic framework for solving the problem, incorporating truncated column generation (CG), and state space reduction techniques. We address the real-time MDVTRSP not only with a focus on providing good plans for the transit operator, in a very efficient way, but also simultaneously accounting for the passenger demand and introducing minimum delay in the involved trips to be serviced. In our solution method, we reschedule the whole disrupted transit network, considering the set of occurred disruptions. The vehicle rescheduling network is rebuilt and resolved, allowing that all no-initiated trips might be rescheduled to a new vehicle route. The heuristics includes the consideration of schedule disruption costs, minimizes the number of changes in the initial off-line scheduling, handles multiple trip disruptions, and deals with thousands of trips and vehicles. The performance of the heuristic algorithm was assessed on randomly generated instances up to 2500 trips, 8 depots, and 3 vehicle types. Computational comparisons showed that the developed algorithm has a potential to be used in real-world applications, giving the low required CPU times.

The contributions of this paper are as follows: (i) to introduce a formulation for the MDVTRSP; (ii) to develop an integrated heuristic framework, combining several modeling and solution techniques, offering very quick solutions for the problem; (iii) to analyze the performance of the developed algorithm variants that compose the heuristic framework,

offering some guidance in which it is the best option for different circumstances; and (iv) to take into account heterogeneous fleet, a common characteristic in transit networks of most large size cities around the world (ROMAN, 2012), but neglected in the scheduling literature (GUEDES; BORENSTEIN, 2015). To the best of our knowledge, the developed approach is the only alternative to solve the real time MDVTRSP, without considering strong assumptions such a depot per each vehicle type and son on.

The paper is organized as follows. Section 2 reviews the literature on the MDVTRSP. Section 3 introduces the problem, explicitly defining our assumptions to model and solve it. The developed mathematical formulation of the problem in presented in Section 4. The solution method is described in detail in Section 5. Section 6 describes the computational experiments carried out to evaluate the performance of the developed solution method. Finally, a summary of the results, and areas of future research are provided in Section 7.

2 Literature Review

The interest in automatic recovery of public transportation systems has followed the development of new information technologies (e.g., cellular phones, global positioning system, geographical information systems). As real-time information is now available at low-cost, transportation companies might react to unexpected events in real time, using automatic recovery tools. Several algorithms were developed for disruption management in airline (CLAUSEN et al., 2010) and railway (CACCHIANI et al., 2014) transportation. In bus-based public transportation, because of the smaller costs involved, the disruption management is still relatively unexplored (VISENTINI et al., 2014). Only very recently, this problem has been receiving some attention in the literature. Huisman, Freling e Wagelmans (2004) was one of the first research studies published in the subject, dealing with robust vehicle scheduling problem, emphasizing delays in the network. Li, Mirchandani e Borenstein (2007), Li, Mirchandani e Borenstein (2009) considered real-time recovery in response to breakdowns in bus passenger transportation, defining the vehicle rescheduling problem, in which vehicles were reassigned to trips. The developed approaches were specially designed to the single-depot setting. Recently, Carosi et al. (2015) developed a decision tool to handle delay and small disruptions, employing a tabu-search procedure for the on-line vehicle scheduling and a CG approach for the consequential crew re-scheduling. However, the tool was developed to be used in a single bus line, restricting the complexity of the problem and its application for a complete transit system.

Some studies were developed to deal with VRSPs in the context of cargo transportation (SPLIET; GABOR; DEKKER, 2014), and integrated rail and bus service networks (JIN; TEO; ODONI, 2015). The former modeled the VRSP based on the capacitated vehicle routing problem. The latter study developed an optimization approach that responds to urban transit rail networks by introducing smartly designed bus bridging services. In a sense, both studies dealt with different problems than the one considered in this research study, focusing on the real-time recovery of bus passenger systems.

Very few published research addresses the MDVTRSP. Dávid e Krész (2014) developed two search algorithms to solve the MDVTRSP. However, they simplified the problem considering that vehicles can be classified into depots depending on two characteristics: the vehicle type, and its starting location at the beginning of the day. Based on these simplifications, they were able to solve a disruption in the transit system as a single-depot problem, justifying the efficiency the developed approach. Zero gap solutions were obtained in less than 0.05s for instances up to 800 trips. For large instances (around 2700 trips), solutions were obtained in less than 15s. Unfortunately, the adopted simplifications are quite

strong and do not reflect the reality of several real world problems, in which more than one operator, with different vehicles in the same depot, serve a transit system. Yıldız (2011) and Uçar, Birbil e Muter (2016) developed robust approaches for managing disruption in the MDVRSP. They considered two disruptions types, namely trip delays and excess of demand in some trips. The developed approach consisted of generating alternatives solution by swapping two planned routes, and incorporate them in a two phase iterative column-and-row generation strategy.

We can conclude that the work on MDVTRSP is rather limited, focusing either on simplified problems or in robust scheduling. The modeling framework developed in this study presents a different approach, focusing on real-time recovery by a complete rescheduling, if necessary, of the transit network, considering the occurrence of a set of disruptions. As a consequence, the MDVRTSP problem defined in this study is more ambitious than the previous developed ones, because it simultaneously considers real-time, an uncompromisable multi-depot setting and heterogeneous fleet.

3 Problem Description

The MDVTRSP can be defined as follows. Given a set of vehicle types with significantly different attributes; a set of depots with their vehicle type capacity; a set of disruptions characterized by a breakdown time and a breakdown local; a set of trips with fixed peak demand, and starting and ending times; given the travel times between all pair of locations, find a feasible minimum-cost scheduling in which (i) each vehicle performs a feasible sequence of trips, starting and ending in the same depot; (ii) the peak demand of each trip is always attended by a vehicle; and (iii) the capacity of each depot for each vehicle type is respected. Before defining a mathematical formulation for the problem, we first describe the problem, introducing its requisites and assumptions taken to solve the problem.

For completeness' sake, we first introduce some definitions and notation to describe the MDVTRSP, following (LI; MIRCHANDANI; BORENSTEIN, 2007). We are given a set of ST service trips. Let S be the set of terminals or stations; K be the set of depots; F the set of vehicle types; and T the planning time horizon. Trip $i \in ST$ can be defined as a n-tuple $(st_i, et_i, ss_i, se_i, IC_i)$, where $st_i \in T$ and $et_i \in T$ are the starting and ending times of trip i, respectively, $ss_i \in S$ and $es_i \in S$ are the starting and ending stations of trip i, respectively, and $IC_i = \{(j, \beta)\}$ is the set of itinerary compatible trips $j \in ST$ with trip i from point β until the ending station of i, given that $et_j - et_i \leq t$. The definition of β depends on the Trips i and j are a compatible pair of trips if the same vehicle can reach the starting point of trip j after it finishes trip i. A route is a sequence of trips in which all consecutive pairs of trips in the sequence are compatible. 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 (BP_i) is the point where trip i is disrupted in time BT_i. Current trip is the trip on which a vehicle is running. It includes both service, waiting, and deadheading (movement of a vehicle to a destination without serving any passenger) 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 may be referred to as the backup trip.

A vehicle can suffer a breakdown in six different situations as follows:

1. while in a service trip;
2. while deadheading between two service trips;
3. while deadheading from a depot to a future service trip;

4. while deadheading from a service trip to a depot;
5. while in a waiting trip in a terminal;
6. while in a waiting trip in a depot.

In the first case, the vehicle is serving passengers. The solution includes sending a backup vehicle to the breakdown point, and from there completing the cut trip, serving the passengers. We assume that an alternative backup vehicle should first complete its current trip before being rescheduled. In all remaining cases, we do not need to pick up the passengers in the disrupted vehicle since it is empty. The solution is to assign a backup vehicle to the starting location of the next trip of the disrupted vehicle. For all considered situations, it is very likely that the solution of the problem provides new routes (a reassignment) for a subset of the pre-assigned vehicles. Also, we can expect some delays in the cut trip in the first five situations.

From the viewpoint of the cut trip, the remaining trips can be divided into the following categories: (i) finished trips; (ii) unfinished trips that have compatible itineraries with the cut trip from the breakdown point; (iii) the remaining unfinished trips; (iv) and trips which starting time is higher than the breakdown point ($et_j > BT_i, \forall j \in ST$) . Trips in situation (i), (ii), and (iii) cannot be rescheduled. Trips in (iv) constitute the set of possible rescheduling trips. Alternative backup vehicles are the ones assigned to trips in situation (ii) and (iv). Only vehicles in depot or serving itinerary compatible trips with the disrupted one are possible alternative backup vehicles.

If a service trip is cut, a preference for backing up the disabled vehicle will be given to a vehicle, if it exists, in a compatible itinerary trip with the disrupted one from the breakdown point. A capacity problem might appear in this case. It is quite possible that some passengers remain in the disabled vehicle that was servicing the cut trip. If the number of passengers remaining on the cut trip is greater than the vacant capacity of the backup vehicle, it is possible that more than one vehicle needs to be sent to the breakdown point. The first vehicle to arrive at the breakdown point picks up some passengers, the next vehicle picks up some more passengers, and so forth until all passengers from the cut trip are served. An empty vehicle from the depot is only sent for capacity issues in situation (i). In that case, we only send a backup vehicle with enough capacity to pick up the passengers from the disabled vehicle. In addition to the capacity problem, we need to consider time constraints related to the travel times of vehicles on current trips. It is not possible to select a vehicle (or vehicles) from an itinerary compatible trip if it has already passed the breakdown point when the disruption occurs or with a very high time interval to reach the disrupted point.

Vehicle rescheduling needs to be solved quickly, since disruption delays will have a negative influence on system performance, since they simultaneously result in increasing costs and deterioration of the level of service. Based on this undesirable snow ball effect, the transit operator manages are, in general, willing to favor efficiency over optimal or close to optimal solutions. Moreover, crew members should be notified as quickly as possible of a new schedule. However, it is very difficult to define an upper bound for the time to obtain a new schedule, since real-time operational is highly context dependent. Nevertheless, computational time requirements may be somewhat mitigated by following a strategy where vehicles currently serving regular trips can only change their routes after finishing their current trips. Li e Head (2009) considered in a single depot context that vehicles not assigned to service trips could be backup ones, defining them as pseudo-depots with vehicle capacity equal to one. The resulting underlying network had a significantly increase in the number of nodes and arcs. Unfortunately, in the multi-depot context, this realistic assumption becomes very difficult to handle, since it would be necessary to create an excessive number of pseudo-depots (stations) and their corresponding trips. Moreover, we need to trace each vehicle in the network. Both issues make the problem extremely difficult to solve even for small instances. Further, the rescheduling implementation would become extremely complex for the human operators and crew involved, raising operational risks probabilities. To keep the problem as simple as possible, allowing a reasonable target time of rescheduling of no more than few minutes (even for large instances), we assume the following in our MDVTRSP formulation:

- Scheduled trips, except of course the cut trip, cannot suffer delays;
- There are restrictions on the number of trips that may be reassigned;
- Only vehicles in the depot or in an itinerary compatible trip with the cut trip are possible backup vehicle candidates.
- There are available technologies for communication between crew members and the operation center of the public transportation operator.

4 Modeling the MDVTRSP

4.1 Network Structures

In this paper, we work with two different vehicle scheduling networks. For modeling the problem, we employed the time-space network (TSN) as the underlying vehicle rescheduling network. However, for solving the problem, we turn this network into a more conventional connection network (CN). On the one hand, in a TSN, nodes represent a specific location in time and space, and each arc corresponds to a transition in time and possibly space. The advantage of this network is the On the other hand, a connection network

A TSN can be defined as a directed graph $TSN = (N, A)$ with N as the set of nodes and A as the set of arcs. Each node $n \in N$ represents a specific location at a certain time and must be determined by reference to $s \in S$. Let $o(k) \in S$ and $d(k) \in S$ denote the same depot $k \in K$, where $o(k)$ means the depot as a vehicle' starting point, and $d(k)$ as its terminating point. The complete set of stations is represented by $CS = S \cup \{o(k), d(k) | k \in K\}$. The set of nodes can be defined as $N = \{(l, t) \,|\, l \in CS, t \in T\}$. The set of arcs A, representing trips, is divided into six subsets, $A = A_{se} \cup A_{wait} \cup A_{dh} \cup A_{pin} \cup A_{pout} \cup A_c$. A_{se} is the set of service arcs, representing a vehicle carrying passengers. Each service arc is defined based on a station and a departure time of a trip i, (ss_i, st_i), and a station and an arrival time of the same trip i, (es_i, t_j). A_{wait} is the set of waiting arcs, representing transitions which a vehicle is waiting at a station to carry out a new trip. A_{dh} is the set of deadheading arcs, representing empty (without passengers) movements of vehicles between two compatible pair of trips. A^{pin} is the set of pull-in arcs, expressing the arcs from depot d to a station $s \in S$, while A_{pout} is the set of pull-out arcs, expressing the arcs from a station $s \in S$ to a depot d. A_c is the set of circulation arcs, where each arc of this set corresponds to the flow from the last to the first depot node. A circulation arc is an artificial one, and serves the purpose of counting the number of performed vehicles blocks. A vehicle route in a TSN is a sequence of n arcs $(a_1, a_2, \ldots, a_{n-1}, a_n), a_i \in A, \forall i$, in which $a_1 \in A_{pin}, a_{n-1} \in A_{pout}, a_n \in A_c$. A complete formal mathematical definition of a TSN can be found in (GUEDES; BORENSTEIN, 2015).

When heterogeneous fleet and multi-depots are simultaneously considered, we construct one network layer for each vehicle type and each depot. A TSN for the MDVTSP is then defined as $TSN^{kf} = (N^{kf}, A^{kf})$, $k \in K$, $f \in F$. The networks overlay by vehicle type and depot, enabling service trips to be in multiple layers.

A connection network for the MDVTSP is an acyclic digraph (V^{kf}, E^{kf}), $k \in K$,

$f \in F$ with nodes $V^{kf} = \{o(k), d(k)\} \cup ST$, and arcs $E^{kf} = \{E^f \cup (o(k) \times ST) \cup (ST \times d(k)\}$, where $E^f = \{(i,j) | st_j \geq et_i + \gamma_{ij}, q_f \geq d_i, q_f \geq d_j, i, j \in ST\}$ is the set of deadheading trips, $(o(k) \times ST)$ is the set of pull-out arcs from depot k, and $(ST \times d(k))$ is the set of pull-in arcs to depot k. A connection network for the MDVTSP can be represented by its adjacent matrix $\mathbf{MDVTSP}^{kf}$, a matrix of costs $(n+m) \times (n+m)$ where n is the number of trips and m is the number of depots. $MDVTSP^{kf}[i][j] = c_{ij}^{kf}$, if $(i,j) \in E^{kf}$, and $MDVTSP^{kf}[i][j] = -1$ otherwise.

On the one hand, the TSN offers space/time information that allows to reduce substantially the number of deadheading arcs in comparison with the CN (STEINZEN et al., 2010), reducing the size of the network. Taking advantage of the explicit representation of time in this network, Kliewer, Mellouli e Suhl (2006) and van den Heuvel, van den Akker e Niekerk (2008) developed specially design procedures to eliminate irrelevant deadheading and waiting arcs from a TSN. The number of deadheading arcs on a TSN becomes $O\big(mn\big)$, in comparison to $O\big(n^2\big)$ of the CN, where m is the number of terminals and n the number of trips. As in general, $n >> m$, this is a considerable reduction for large instances of the problem. We decided to use this advantage to represent and to formulate the problem, employing a TSN as the underlying vehicle rescheduling network, following the strict computational efficiency requisite of the MDVTRSP. On the other hand, the CN offers a more convenient computational representational towards solving the MDVSP (RIBEIRO; SOUMIS, 1994; GUEDES; BORENSTEIN, 2015). We applied he algorithm introduced in (GUEDES; BORENSTEIN, 2015) to convert a TSN into a CN. This algorithm transforms a TSN arc $a \in A_{wait} \cup A_{dh} \cup A_{pin} \cup A_{pout}$ in nodes in the CN, representing the trips involved in this arc, and deadheading arcs (including pull-in and pull-out arcs, when one of the nodes is a depot) connecting the two associated nodes. Waiting arcs in the TSN are represented as deadheading arcs in the CN. In this algorithm, only deadheading and waiting arcs which its initial node represents the end of a service trip and its final node is the starting node of a service trips are considered, further reducing the size of the CN generated.

4.2 Vehicle Rescheduling Network

In this section, we define the vehicle rescheduling network given the disruption of trip r in time BT_r, local BP_r, and estimated number of passengers PAX_d. As previously stated all trips $j \in ST$ with starting time $st_j \leq BT_r$ cannot be rescheduled. Thus, the set of services trips to be rescheduled can be defined as $ST^R = \{j \in ST | st_j > BT_r\}$. The new rescheduling depends on the vehicle situations as aforementioned in section 3.

If a service trip is disrupted (condition 1), we need to send a back vehicle to pick up the passengers. If there are no compatible itinerary trips capable of serving all

passengers, we need to include a new trip r in the set of service trips to be rescheduled, starting in the breaking point BP_r, and with a starting time $st_r = BT_d + \delta_t$, where δ_t is an acceptable time for the backup vehicle arrive in the disruption point. The value of δ_{t1} is the maximum delay of the cut trip. The ending point of trip r is the same of trip d, and $et_r = et_d + \delta_{t1}$. The set of rescheduling trips is defined as $RT = ST^R \cup \{r\}$. A new station is added to the set of stations, as it is considered as a starting local of a this new service trip to be rescheduled. Let $CS^R = S \cup \{BP_d, o(k), d(k) | k \in K\}$ be the set of stations in the rescheduling network.

If the disrupted vehicle is either in deadheading (including pull-in and pull-out trips) or in a waiting trip, related to conditions 2 to 6 described in section 3, we need only to check the next trip in the vehicle route. If the next planned scheduling trip $j \in ST$ has a $st_j < BT_d + \delta_{t2}$, we altered the starting and ending times of this trip to $st_j = BT_d + \delta_2$ and $et_j = BT_d + \delta_2$, respectively, and update set ST, accordingly. Otherwise, set ST remains unchanged. We then define $RT = ST^R$ and $CS^R = CS$. δ_{t2} is also the maximum delay allowed for the cut trip. In general, $\delta_{t2} \leq \delta_{t1}$.

The rescheduling TSN can be defined as a directed graph $TSN^R = (N^R, A^R)$, with nodes $N^R = \{(l,t) \,|\, l \in CS^R, t \in [BT_d, T]\}$, and arcs $A^R = A^R_{se} \cup A^R_{wait} \cup A^R_{dh} \cup A^R_{pin} \cup A^R_{pout} \cup A^R_c$. These sets can be formally defined as follows. Let $A^R_{se} = \{((ss_i, st_i), (es_i, et_i)) \in N^R \times N^R\}$ be the set of service trips $i \in RT$ that can be rescheduled. Let $A^R_{wait} = \{((es_i, et_i), (ss_j, st_j)) \in\in N^R \times N^R | es_i = ss_j, st_j > et_i\}$ be the set of waiting trips between two trips $i, j \in RT$. Let $A^R_{dh} = \{i_e = (es_i, et_i), j_s = (ss_j, st_j) \in N \times N \,|\, es_i \neq ss_j, st_j \geq et_i + \gamma_{ij}\}$ be the set of deadheading arcs among service trips $i, j \in RT$, where station of trip j. Sets A^R_{pout}, A^R_{pin}, and A^R_c follows the aforementioned definition, properly considering nodes in N^R.

4.3 Mathematical Formulation

Given the $TSN^{Rkf} = (N^{Rkf}, A^{Rkf})$, $\forall k \in K, \forall f \in F$, the MDVTRSP can be formulated as an integer linear programing model. In the MDVTRSP context, an arc in the deadheading set should respect not only time compatibilities, but the capacity of a vehicle, e.g., the peak demand of a trip must be smaller than the capacity of the vehicle assigned to perform this trip. The main input of the MDVRSTP is the initial scheduling, represented by $TSN^{Skf} = (N^{Skf}, A^{Skf}), k \in K, f \in F$, where $A^{Skf} \subset A^{kf}$, that contains only the selected vehicle routes in the off-line planning. Before stating a formal formulation for the MDVTRSP, we first introduce some additional parameters. Let c^{kf}_{ij} be the cost of a vehicle type f from depot k being assigned to arc $(i,j) \in A^{Rkf}$. Let d_i be the peak demand of trip i. Let q_f be the capacity of vehicle f. Let v^k_f be the number of vehicles type f in depot k. Let p_{ij} be a penalty cost if a new deadheading arc $a^{Rkf}_{ij} \notin A^{Skf}$ is included in the rescheduling solution. We now introduce the decision variables. Let x^{kf}_{ij} be

a binary decision variable, with $x_{ij}^{kf} = 1$ if a vehicle type f from depot k is assigned to trip j directly after trip i, and $x_{ij}^{kf} = 0$ otherwise. The MDVTRSP can be formulated as follows:

$$\min \sum_{k\in K}\sum_{f\in F}\sum_{(i,j)\in A^{Skf}} c_{ij}^{f} x_{ij}^{kf} + \sum_{k\in K}\sum_{f\in F}\sum_{(i,j)\in A^{Rkf}\setminus A^{Skf}} (c_{ij}^{f} + p_{ij}) x_{ij}^{kf} \tag{4.1}$$

s.t.

$$\sum_{f\in F}\sum_{k\in K}\sum_{j\in(i,j)\in A^{Rkf}} q_f x_{ij}^{kf} \geq d_i \qquad \forall i \in A_{se}^{Rkf} \tag{4.2}$$

$$\sum_{f\in F}\sum_{k\in K}\sum_{j\in(i,j)\in A^{Rkf}} x_{ij}^{kf} = 1 \qquad \forall i \in A_{se}^{Rkf} \tag{4.3}$$

$$\sum_{j:(o(k),j)\in A^{Rkf}} x_{i,j}^{kf} \leq v_f^k \qquad \forall k \in K, f \in F \tag{4.4}$$

$$\sum_{j:(j,i)\in A^{Rkf}} x_{ji}^{kf} - \sum_{j:(i,j)\in A^{Rkf}} x_{ij}^{kf} = 0 \qquad \forall i \in L^{kf}, \forall k \in K, f \in F \tag{4.5}$$

$$x_{ij}^{kf} = \{0,1\} \qquad \forall (i,j) \in A^{Rkf}, k \in K, f \in F \tag{4.6}$$

where $L^{kf} = \{(l,t) | l \in S, t \in T\}$. Clearly, $L^{kf} \subset N^{Rkf}$. The objective function (4.1) minimizes simultaneously the total operational costs and changes in the off-line scheduling. Constraints (4.2) guarantee that the peak demand of every service trip is respected. The constraints (4.3) ensure that each service trip is executed exactly once for a vehicle. Constraints (4.4) respect the number of vehicles of each type that can be used from every depot, while (4.5) are flow conservation constraints. The domain of the decision variables is defined by constraints (4.6).

Guedes e Borenstein (2015) showed that the MDVTSP is an NP-hard problem, considering the off-line network TSN^{kf}. In the worst case scenario, a disruption could occur very early in the planning horizon, making the rescheduling network dimension similar to the off-line network, e.g., the $\left|A_{se}^{Rkf}\right| \approxeq \left|A_{se}^{kf}\right|$. As a consequence, the MDVTRSP is also an NP-hard problem.

5 Solving the problem

In this section, we propose a heuristic method to solve the problem, based on the heuristic framework introduced by Guedes e Borenstein (2015) to solve the MDVTSP.

5.1 Solution Approach

The overall heuristic approach for the MDVTRSP is presented in Algorithm 5. The solution method prioritizes efficiency over efficacy, following the problem requisites previously discussed in section 3. The main contribution of the solution method is to carry out very quickly a complete rescheduling of the transit system, given that a serious disruption occured. The method considers all no-initiated trips in the rescheduling, rebuilding the underlying scheduling network, and solving it by integrating state space reduction techniques and truncated CG. The state space reduction procedure aims to reduce the set of variables to a smaller, but relevant, subset of variables, decreasing the complexity of the instances. The CG approach solves the reduced state space problem using an improved truncated approach towards accelerating the CG stabilization, including the use of an initialization procedure based on the routes of both the planned scheduled and solution of the state space selection procedure.

5.2 Finding Itinerary Compatible Trips

Algorithm 6 describes the developed procedure for finding a set of itinerary compatible trips with the disrupted trip r, characterized by variables BP_r, BT_r, and PAX_r. The routine assumes that the vehicle crew members can exchange information with the operation center of the public transport operator anytime, such as the current number of passengers in the vehicle, and location. The function returns the set of itinerary compatible trips that have available capacity to get all or some passengers of the cut trip. If all passengers can be served by the vehicles assigned to itinerary compatible trips set ICT, the value found $= 1$ is returned and no rescheduling is required.

5.3 State Space Reduction

Although the rescheduling network has been optimized in terms of eliminating irrelevant deadheading and waiting arcs, the MDVTRSP is still very difficult to solve efficiently for very large instances. Experiments conducted by Pepin et al. (2009) and Otsuki e Aihara (2014) with the MDVSP showed that state-of-the-art algorithms required

Algorithm 5 Solution approach overview

Input: A cut trip r with a disruption time (BT_r) and local (BP_r). The planned schedule and estimates of the peak demand of the service trips.
Output: A new rescheduling plan that simultaneously minimize the operational costs and changes in the original plan.

Step 1 **(Initialization):** Set δt_1 and δt_2.

Step 2 **(Compatible Itinerary Trips):** Find backup vehicle(s) from itinerary compatible trips to r from BP_r using Algorithm 6. If ($found = 1$) then trips in set ICT can recover all passengers in the cut trip. Stop. Otherwise go to Step 3.

Step 3 **(Vehicle Rescheduling Network):** Generate the rescheduling time-space network (RTSN), considering a set of cut trips and the values of δt_1 and δt_2, based on the off-line timetabling and peak demand of trips.

Step 4 **(TSN Reduction):** Reduce the RTSN by applying the reduction procedure suggested by Kliewer, Mellouli e Suhl (2006).

Step 5 **(Network Conversion):** Convert the rescheduling time space network to a rescheduling connection network (RCN), using the algorithm introduced by Guedes e Borenstein (2015).

Step 6 **(State Space Reduction):** Reduce the RCN to obtain a reduced rescheduling connection, by applying the state space selection described in Section 5.3.

Step 7 **(Initial Solution Generation for the CG):** Set all paths obtained in the solution of the state space selection procedure obtained in Step 5, and the paths in the off-line scheduling (Ω^{Skf}) into a set of initial columns.

Step 8 **(Column Generation):** Apply Algorithm 7 in the RCN, using the set of columns defined in Step 7 as initial solutions.

Algorithm 6 Procedure to find itinerary compatible trips

$ICT \leftarrow \emptyset$, $found \leftarrow 0$, $pax \leftarrow PAX_r$, $A \leftarrow \emptyset$
for all $i \in IC_r$ **do**
  Obtain the position, the time to arrive to BP_r (t_{ir}), and available capacity (cap_i) of each vehicle assigned to i
  if (($t_{ir} \leq \delta t_1$) and ($cap_j > 0$) **then**
    $A \leftarrow A \cup \{i\}$
  end if
end for
if $A \neq \emptyset$ **then**
  Sort trips in ICT in crescent order based on t_{ir}
  $j \leftarrow FIRST(ICT)$
  while (($pax \leq 0$) and ($j \neq$ NULL)) **do**
    $pax \leftarrow pax - cap_j$
    $ICT \leftarrow ICT \cup \{j\}$
    $j \leftarrow NEXT(ICT)$
  end while
  if $pax \leq 0$ **then**
    $found \leftarrow 1$
  end if
else
  Stop
end if

around 780 and 2800 seconds to solve instances with 8 depots, and 1000 and 1500 trips, respectively. These CPU times are incompatible with the rescheduling problem. In order to speed up the solution process, we employ the state space selection procedure developed by Guedes et al. (2015). The main objective of this procedure is to reduce the complete state space to a simpler, more manageable representation by focusing attention on a selected set of deadheading trips. The procedure further eliminates deadheading trips with very small possibility of being selected in the final solution, decreasing the number of variable to be solved by the CG algorithm.

In vehicle scheduling problems, it is natural that several deadheading trips are very expensive to be carried out, since trips that are close geographically can be far apart due to times in which they must start, while others may be compatible in terms of timing, but geographically infeasible. Several deadheading trips become too costly to be considered as viable alternatives in an optimal or close to optimal solution, both due to distance or timing issues. As a consequence, although the complete solution space is very large, a small number of variables x_{ij}^{kf} has a high chance of being used in the final solution. The goal of this procedure is to identify the set of relevant variables that are selected by any solution of all $|F| \times |K|$ single-depot VSPs (SDVSPs). The reasoning behind this procedure is quite intuitive. If a variable is not chosen as a solution considering $|F| \times |K|$ individual SDVSPs, then it would have small chance to be chosen as a candidate solution when considering the whole problem.

The procedure consists in efficiently solving $|F| \times |K|$ individual SDVSPs. The arcs found in any of the solved SDVSPs configures a reduced connection graph, represented by matrix $RMDVTRSP^{Rkf}$, with $RMDVTSP^{Rkf}[i][j] = MDVTSP^{Rkf}[i][j]$ if arc (i, j) is in the solution of any $SDVSP^{kf}, k \in K, f \in F$; and $RMDVTSP^{Rkf}[i][j] = -1$, otherwise. It should be noted that pull-in and pull-out arcs cannot be reduced since they introduce severe instabilities in the CG application. Each SDVSP problem is structured by matrix VSP^{Rfk}, easily obtained from $MDVTRSP^{Rkf}$ by taking into consideration only the pull-in and pull-out trips to and from depot k, respectively, and vehicle type f. Each SDVSP can be modeled as an assignment problem (AP) following Paixão e Branco (1987), and solved using algorithm LAPJV (JONKER; VOLGENANT, 1987), a well praised solution method for the AP (DELL'AMICO; TOTH, 2000).

5.4 Initial Solution for the CG

An initial solution is needed in CG. The use of artificial variables penalized by a big-M cost is the usual approach (PEPIN et al., 2009). However, when applied to the MDVTRSP in initial experiments, the CG was taking excessive CPU time to find a solution where all artificial variables are zero. This was caused due to the well known *heading-in*

effect in CG of initially producing irrelevant columns (LÜBBECKE; DESROSIERS, 2006). The literature describes several methods developed towards better initialization in CG (VANDERBECK, 2005). However, since there is no guarantee that good initial solutions will improve the convergence process, we decided to apply intuitive and almost free-CPU time approaches.

Firstly, we considered all paths in the solution of the off-line scheduled as initial solutions. Let Ω^{Skf} be the set of off-line scheduled paths by vehicle $f \in F$ from $o(k)$ to $d(k), k \in K$. If a path $p \in \Omega^{Skf}$, then the correspondent variable θ_p is set to one, and included in the set of initial columns θ_{init}. Initial experiments showed that some improvements were obtained, but the heading-in effect was still present for some instances.

We decided then to incorporate an additional procedure, taking into advantage of the several paths obtained by the state space selection procedure. We generate additional initial solutions using the paths p obtained by the state space selection procedure described above. If a path p is in the solution of any of the $|F| \times |K|$ solved SDVSPs, then this path is inserted in Ω^{Red}, and becomes an additional column in θ_{init}.

Set θ_{init} is then used to compute an initial solution for the restricted master problem (RMP) in the CGM, offering an upper bound on the integer optimal value. Its dual solution provides a lower bound on the RMP. Preliminary experiments show that these two bounds are extremely helpful for speeding up the convergence process of the CG with an average loss of around 5% in the value of the best solution obtained without using this scheme. It seems that these two bounds create an effect that is similar to stabilization processes described in Lübbecke e Desrosiers (2006), significantly smoothing the dual variables convergence towards their optima. Although very simple in comparison with previous developed approaches, it has the advantage of being simultaneously very effective and efficient. When used in conjunction with the state space selection procedure, the additional CPU time required to handle the initial set of columns is insignificant, being suitable to be used in the real-time MDVTVRSP.

5.5 Truncated Column Generation

In this section, we propose a truncated CG algorithm for solving the MDVTRSP. The approach developed in this paper is based on the algorithm presented in Guedes e Borenstein (2015) for solving the MDVTSP.

5.5.1 Primal Problem

The MDVTRSP can also be formulated as a set partitioning problem, based on the formulation of Ribeiro e Soumis (1994) for the MDVSP. Let $G^{Skf} = (V^{Skf}, E^{Skf}), k \in$

$K, f \in F$ be the planned scheduling CN. Let $G^{Rkf} = (V^{Rkf}, E^{Rkf}), k \in K, f \in F$ be the rescheduling CN obtained from TSN^R. Let Ω^{Rkf} be the set of all possible rescheduling paths by vehicle $f \in F$ from $o(k)$ to $d(k), k \in K$. Let $p \in \Omega^{Rkf}$ be a vehicle route or path in the rescheduling network. Let P be a penalty cost if a new path $p \in \Omega^{Rkf} \setminus \Omega^{Skf}$ is selected in the rescheduling. Let c_p the cost of executing route p. Let $a_{ip} = 1$ if trip $i \in V^{Rkf}$ is covered by route p, and $a_{ip} = 0$ otherwise. Let binary variable θ_p be 1 if route p is selected in the solution, and $\theta_p = 0$ otherwise. The path-based formulation for the MDVTRSP is as follows:

$$\min \sum_{f \in F} \sum_{k \in K} \sum_{p \in \Omega^{Skf}} c_p \theta_p + \sum_{f \in F} \sum_{k \in K} \sum_{p \in \Omega^{Rkf} \setminus \Omega^{Skf}} (c_p + P)\theta_p \tag{5.1}$$

s.t.

$$\sum_{f \in F} \sum_{k \in K} \sum_{p \in \Omega^{Rkf}} a_{ip}\theta_p = 1, \qquad \forall i \in V^{Rkf} \tag{5.2}$$

$$\sum_{f \in F} \sum_{k \in K} \sum_{p \in \Omega^{Rkf}} q_f a_{ip}\theta_p \geq d_i \qquad \forall i \in V^{Rkf} \tag{5.3}$$

$$\sum_{p \in \Omega^{Rkf}} \theta_p \leq v_{kf}, \qquad \forall k \in K, \forall f \in F \tag{5.4}$$

$$\theta_p \in \{0, 1\} \qquad \forall p \in \Omega^{Rkf} \tag{5.5}$$

Objective function (5.1) aims to simultaneously minimize the total operational costs and possible changes in the off-line planned paths. Constraints (5.2) ensure that each trip is performed by exactly one vehicle. Constraint (5.3) guarantee that the peak demand of each trip is satisfied, while constraints (5.4) assure that the depot capacity for each vehicle type is respected.

The relaxation of model (4.1) – (4.6) constitutes the RMP of the CG approach. As the number of paths in G^{Rkf} can still be huge, mainly if a disruption occurs in the beginning of the planning horizon, the RMP is hard to solve. Let $RG^{Rkf} = (V^{Rkf}, RE^{Rkf}), k \in K, f \in F$ be the reduced rescheduling CN obtained by using the selection state space reduction process in network G^{Rkf}. Observe that only arcs are reduced, therefore the two networks have the same set of nodes V^{Rkf}. Let Ω_r^{Rkf} be the be the set of all possible rescheduling paths in RG^{Rkf}. If we replace set Ω^{Rkf} by Ω_r^{Rkf} in the objective function and in all constraints of the RMP, we obtain a new problem, called the reduced restricted master problem (RRMP). In the RRMP, the number of possible paths is significantly smaller than in the RMP, consequently reducing the number of variables θ_p. In order to speed up the solution process, the MDVTRSP is solved by dynamically generating the paths, based on a Dantzig-Wolfe decomposition, through CG by repeatedly solving (i) the RRMP with a subset of columns and (ii) a pricing subproblem to produce columns with negative reduced cost. Even if the RRMP is solved to optimality by the CG algorithm, it is not guaranteed that the resulting optimal solution is integral. A simple rounding procedure was included in the CG algorithm to provide integer solutions as better explained at

Section 5.5.3).

5.5.2 Pricing Problem

Let dual variables γ, π, and σ correspond to constraints (4.2), (4.3), and (4.4), respectively. Considering these dual variables, we have the following pricing problem on the reduced network RG^{Rkf}:

$$\min \; -\sigma_{kf} + \sum_{(i,j)\in RE^{Rkf}} (c_{ij}^{f} - \pi_j - q_f\gamma_i)x_{ij}^{kf} \tag{5.6}$$

s.t.

$$\sum_{j:(j,i)\in RE^{Rkf}} x_{ji}^{kf} - \sum_{j:(i,j)\in RE^{Rkf}} x_{ij}^{kf} = 0 \qquad \forall i \in W^{Rkf}, \forall k \in K, f \in F \tag{5.7}$$

$$x_{ij}^{kf} \in \{0,1\} \qquad \forall (i,j) \in RE^{Rkf}, k \in K, f \in F \tag{5.8}$$

where $W^{Rkf} = V^{Rkf} \setminus \{o(k), d(k)\}$.

The pricing problem consists of finding the shortest path from origin depot $o(k)$ to same destination depot $d(k)$ in the RG^{Rkf} network. This problem can be easily solved by a specialized algorithm.

5.5.3 Modified Truncated CG Algorithm

The developed truncated CG as based in the algorithm developed by Pepin et al. (2009). This algorithm requires three predefined parameters, namely Z_{min}, I, and Ω_{min}. The algorithm terminates early if the optimal value of the (R)RMP has not decreased by more than Z_{min} in the last I iterations. Parameter Ω_{min} is a threshold value in the rounding of variables θ_p. However, we introduced two changes to this algorithm based on Guedes e Borenstein (2015), better customizing it to the MDVTRSP. We introduced a fourth parameter, $Num\_Stabilized < I$. This parameter is used to stop generating columns when the objective function of the (R)RMP remains unaltered in the last $Num\_Stabilized$ iterations. A rounding step is then carried out to perturb the solution, avoiding a stagnation of the search process in the same point of the state space solution. This strategy is a good way of reducing the *tailing-off* effect in the CG convergence for the MDVTRSP. The second change was in the way the columns are inserted in the (R)RMP. Our approach uses the traditional way of selecting columns to insert in the current master problem. However, it solves the master problem right after a new column is inserted. The idea is to select better columns by always using updated duals, avoiding the generation of many similar columns. Algorithm 7 presents an outline of the developed modified truncated CG.

The use of initial solutions, the introduction of a reduced restricted master problem, and the use of these two strategies to avoid the common problems in CG convergence,

Algorithm 7 Modified Truncated Column Generation for the MDVTRSP

Step 1 **(Initialization)**: Set parameters Z_{min}, I_{min}, $Num\_Stabilized$ and θ_{min}. Set $n \leftarrow 1, n_0 \leftarrow 1$ and $stabilized \leftarrow 0$.

Step 2 **Initial Solution** Set the initial set of columns $\theta_{init} = \{\theta_p = 1, \forall p \in \Omega_n^{Rkf}\}$, where $\Omega_n^{Rkf} \leftarrow \Omega^{Red} \cup \Omega^{Skf}$.

Step 3 **(Reduced Restricted Master Problem):** Solve the RRPM. If $(n = 1)$ then use the set of initial columns θ_{init}. Otherwise, use the current set θ_n. Obtain a primal solution $(\theta_{\mathbf{n}}, \delta_{\mathbf{n}})$, and cost Z_n^{RRMP}.

Step 4 **(Stabilization Test):** If $(stabilized > Num\_Stabilized)$ then go to step 7; otherwise go to step 5.

Step 5 **(Early Termination Test):** If $(n - n_0 > I_{min})$ and $(Z_{n_0}^{RRMP} - Z_n^{RRMP} < Z_{min})$ then go to step 7; otherwise go to step 6.

Step 6 **(Pricing Solution):** For each $k \in K$ and for each $f \in F$ do

Step 6.1 Update the costs of arcs A^{Rkf} using dual variables $(\pi_n, \sigma_n, \gamma_n)$. Solve a shortest path in graph RG^{Rkf} using the SLF algorithm (BERTSEKAS, 1993).

Step 6.2 If the subproblem presents a negative reduced cost then set $\Omega_{n+1}^{Rkf} \leftarrow \Omega_n^{Rkf} \cup S_n^{Rkf}$, where S_n^{Rkf} is the solution of the subproblem, $n \leftarrow n + 1$. Solve the RRPM with the updated set of columns. If $Z_{n-1}^{RRMP} - Z_n^{RRMP} = 0$ then $stabilized \leftarrow stabilized + 1$ else $stabilized \leftarrow 0$. Go to step 4.

Step 7 **(Master Problem Feasibility Test):** If $(\delta_{\mathbf{n}} \neq 0)$ STOP; else go to step 8.

Step 8 **(Integrality):** If $(\theta_{\mathbf{n}} \in \{0, 1\})$ then the solution was found. Return θ_n. Stop.

Step 9 **(Variable Fixing):** Perform a rounding of the no-integer variables in $\theta_{\mathbf{n}}$. If $(\theta_{p,n} \geq \theta_{min})$ set $\theta_p \leftarrow 1$. If no such variables exist set the highest value in $\theta_{\mathbf{n}}$ as one. Set $n \leftarrow n + 1$ and $n_0 \leftarrow n$. Go to Step 3.

has created a solution approach that is capable to solve the MDVTRSP up to few CPU minutes even for large instances.

6 Computational Experiments

The main objective of the computational experiments was to evaluate the performance of the developed solution approach, in terms of quality and efficiency of the solution, using random generated instances, comparing some algorithm variants under different parametrization.

6.1 Experimental Setup

Problem instances were generated as in Guedes e Borenstein (2015). First, stations $s_1, s_2, \ldots, s_{20}$ were created as the nodes of a planar graph employing Dorogovtsev e Mendes (2002)'s algorithm. Next, trips were generated. For each trip $i, i = 1, \ldots, n$ the starting and relief stations were randomly selected. The starting time st_i followed a uniformly random integer in interval (0,23). However, as trips are not uniformly distributed during the planning horizon, being more frequent in rush hours, we integrated a further random process to include a trip in the timetabling. The frequency of trips ($FT(st_i)$) was based on a curve obtained by a weighted sum of three Gaussian distributions, with averages of 7, 12, and 18, and standard deviations of 2, 1.5, and 3, respectively. These values were obtained based on exiting timetables of several cities in Brazil. The average values represent rush hours. The weights of each distribution were 4.5, 1, and 5, respectively, guaranteeing that the resulting curve has a total area equal to 1. Along with st_i, an additional uniformly distributed variable, $u(0, 1)$, was generated. Trip i is included in the timetabling if $u \geq FT(st_i)$. Since the ending time et_i of trip i must include a travel time between stations s_a and s_b, and a dwell time at vehicle stops, we generated $et_i = (d(s_a, s_b) + Dist_{min}) * rnd(0, 150) * 60$, where $Dist_{min} = 180$s, and rnd is a function that generates uniformly random integer in interval (0,150). The demand of trip $i, D(i)$, was computed using $D(i) = (rnd(0, Q) * FT(st_i)) + D_{min}$, where Q is the maximum vehicle type capacity, and D_{min} is a minimum demand value. In our experiments, $D_{min} = 20$.

Three different vehicles were considered in the experiments, with capacities 84 (type A), 9 (type B), and 190 (type C) passengers. Travel costs (including waiting costs) c^f_{ij} were defined based on the vehicle type, using cost factors directly related with the vehicle fixed cost. Cost factors (α_f) of 0.8, 1, 1.2 were computed for vehicle types A, B, and C, respectively. The travel costs were computed as follows:

- $c^f_{ij} = \alpha_f * (st_j - et_i) * 8$, for the deadheading trips between two compatible trips i and j.

- $c_{ij}^f = \alpha_f * (st_j - et_i) * 4$, for the waiting trips between two compatible trips i and j.
- $c_{kj}^f = \alpha_f * (5,000 + d(o(k), s_j))$, for the pull-out trips from depot $k \in K$.
- $c_{jk}^f = \alpha_f * (5,000 + d(s_j, d(k))$, for the pull-in trips to depot $k \in K$.

For the comparison of the algorithms, random instances were generated with $ST = \{500, 1000, 1500, 2000, 2500\}$ and $K = \{4, 8\}$. For each combination of trips, depots, and vehicle types, five instances were generated, totaling 200 instances. Given the random nature of the instances, itinerary compatible trips were not considered in the experiments. The solution process is better stressed when this issue is not taken into account.

To evaluate the performance of the developed approach, we first generated a timetabling for each tested instance. Next, an MDVSTP problem was solved. Then, a disruption was randomly introduced at around 6 *a.m.* in one of the trips scheduled to start before this time. We assumed that trips are cut in a distance proportional to the difference between its starting time and the disruption time. For instance, if a disruption occurs in a trip with a duration of one hour after 15 minutes of its starting, the breakpoint point is located at 1/4 of the starting station in the service arc representing this trip.

The following notation is used to indicate the compared algorithms: (i) MTCG: Modified truncated CG with variable fixing, without considering initial solutions or state space reduction. This variant do not consider Steps 6 and 7 of Algorithm 5; (ii) JVCG: In this variant, Step 6 is carried out, but not Step 7, and the CG approach is run with reduced matrix $RMDVTRSP$; (iii) ISCG: Both Steps 6 and 7 are executed, but matrix $MDVTRSP$ is solved by the CG approach; and (iv) IJCG: The complete framework presented in Algorithm 5, using the state space reduction procedure and the initial solutions for the CG approach. The algorithms were implemented in C++. All experiments were carried out in an Intel Xeon CPU E5-1603, 2.80 GHz, 16 GB de RAM. We employed CPLEX 12.5 to solve all mixed linear programming models.

The settings of CG parameters, namely Z_{min}, I, and Ω_{min}, were carried out by experimentation, using some of the instances generated for the computational experiments. Different combinations of these three parameters led to distinct solutions and CPU times. The best quality solutions were obtained with small values of Z_{min} and large values of I and Ω_{min}, but, in general, with low efficiency. The best solutions were found with $Z_{min} = 0$. On the one hand, the solution quality was improved with the increase of parameter I. On the other hand, the solution efficiency was negatively affected by the increase of I. Good compromised solutions, in terms of quality and CPU times, were obtained with $I = 10$. Since our initial experiments were not conclusive in terms of finding a clear correlation between this parameter and the solution quality, we set $\Omega_{min} = 0.5$, a quite low value comparing with the off-line scheduling, towards accelerating the solution process.

6.2 Results

This section presents the obtained results from the experiments carried out, analyzing the most important findings. Table 9 and 10 compare the performance of algorithms MTCG, JVCG, ISCG, and IJCG for five instances of each problem, for values of P equal to 1,000 and 100,000, respectively. The first two columns display the instance dimensions, in terms of the number of trips and depots, respectively. The next column shows the algorithm variant. The next four columns show the average CPU time, the average gap (*gap*), the average speedup factor (Speedup), and the average changes (in percentage) from the off-line scheduling (Exchanges). The average gap (*Gap*) and the speedup factor were computed taken the MTCG as comparison basis. For instance, $Gap = 100 \times \frac{S_A}{S_{MTCG}}$, where S_{MTCG} is the average best solutions obtained by algorithm MTCG, and S_A is the average best solution obtained by the algorithm being compared, say algorithm A. As can we see the $P = 100,000$ display smaller average changes and gaps albeit the speedup decrease a little. For $P = 100,000$ the gap, speedup factor and route change are 1.13%, 9.43, 44.24% for JVCG; 5.62%, 17.71, 54.28% for ISCG and 0.92%, 23.21, 45.44% for IJCG, respectively. So from here the results will be with this parameter value.

As has been pointed out by the table 10, algorithms IJCG, JVCG, and ISCG, in general, offer good solution quality (some with negative gaps) very efficiently. IJCG is the better trade-off algorithm because the gap is under 2% almost all the time and the speedup factor up to 9.5 (500 trips and 8 depots) until 51.90 (2500 trips and 4 depots) and, in average, 23.21.

Based on the results reported by table 10, it is possible to state that the use of state space reduction for the CG algorithm had a significant effect in the efficiency of the algorithms, while the initial solution had a major effect in the quality of the solution.

Analyzing the solution quality, algorithms IJCG, JVCG, and ISCG, on average, presents a gap around 2.56%. For IJCG the maximum gap was 2.23% and 0.92% in average. The average CPU time for all algorithms is highly dependent on the instance size. Overall, our developed implementations were quite efficient. Algorithm IJCG obtained in average the best solutions of all tested algorithms, considering both quality and efficiency. IJCG obtained better quality solutions with a speedup factor of 23.21, on average, in comparison with algorithm MTCG, a more traditional CG implementation for the MDVTRSP. The simultaneous use of good initial solutions and state space reduction significantly increased the convergence process of the CG method, decreasing or eliminating the well-known tailing effect (LÜBBECKE; DESROSIERS, 2006). Table 10 shows that either the use of initial solutions for the truncated CG algorithm or the state space reduction led to efficient algorithms. The use of these procedures decreased, in several instances, the number of required iterations to find a good solution.

Table 9 – Comparison results with $P = 1000$

N	D	Alg	CPU(s)	Gap	Speedup	Route Changes
500	4	MTCG	58.20	-	-	75.43%
500	4	JVCG	8.00	10.16%	7.20	78.53%
500	4	ISCG	49.20	0.00%	1.17	75.74%
500	4	IJCG	3.80	8.78%	16.67	79.41%
500	8	MTCG	85.60	-	-	77.62%
500	8	JVCG	13.00	9.49%	6.66	77.96%
500	8	ISCG	85.00	-0.73%	1.02	77.27%
500	8	IJCG	9.80	9.00%	9.15	79.61%
1000	4	MTCG	542.00	-	-	73.41%
1000	4	JVCG	44.40	9.77%	13.13	78.20%
1000	4	ISCG	89.60	6.40%	25.36	83.21%
1000	4	IJCG	17.00	6.40%	36.83	83.02%
1000	8	MTCG	600.00	-	-	74.49%
1000	8	JVCG	95.40	8.65%	6.31	76.37%
1000	8	ISCG	337.20	4.54%	5.02	76.15%
1000	8	IJCG	55.60	8.52%	11.17	76.51%
1500	4	MTCG	1699.40	-	-	73.00%
1500	4	JVCG	163.80	8.78%	10.72	77.76%
1500	4	ISCG	38.40	10.76%	44.90	83.99%
1500	4	IJCG	49.00	8.78%	38.22	80.88%
1500	8	MTCG	2046.80	-	-	80.60%
1500	8	JVCG	257.60	8.53%	8.07	83.11%
1500	8	ISCG	126.80	11.68%	16.24	85.13%
1500	8	IJCG	132.40	7.86%	15.85	83.03%
2000	4	MTCG	4312.00	-	-	75.43%
2000	4	JVCG	311.40	8.92%	14.04	78.11%
2000	4	ISCG	68.00	9.83%	63.54	83.61%
2000	4	IJCG	60.40	9.51%	71.63	81.77%
2000	8	MTCG	4622.00	-	-	73.90%
2000	8	JVCG	582.00	5.46%	7.99	77.53%
2000	8	ISCG	1383.80	5.40%	18.32	77.12%
2000	8	IJCG	253.20	5.84%	20.12	76.90%
2500	4	MTCG	8680.00	-	-	71.14%
2500	4	JVCG	575.67	8.32%	15.25	76.50%
2500	4	ISCG	114.67	8.47%	76.15	80.92%
2500	4	IJCG	105.33	8.33%	83.82	80.23%
2500	8	MTCG	8637.00	-	-	66.83%
2500	8	JVCG	923.40	7.69%	9.43	71.67%
2500	8	ISCG	807.40	10.90%	25.46	72.07%
2500	8	IJCG	291.40	8.60%	34.80	72.53%

Table 10 – Comparison results with $P = 100000$

N	D	Alg	CPU(s)	Gap	Speedup	Route Changes
500	4	MTCG	53.00	-	-	56.41%
500	4	JVCG	7.40	0.47%	7.20	49.31%
500	4	ISCG	46.40	-1.13%	1.13	57.67%
500	4	IJCG	3.80	-0.62%	14.52	50.15%
500	8	MTCG	81.00	-	-	60.01%
500	8	JVCG	12.20	2.53%	6.68	53.23%
500	8	ISCG	86.00	0.37%	0.94	59.83%
500	8	IJCG	9.20	2.23%	9.51	53.70%
1000	4	MTCG	531.20	-	-	54.79%
1000	4	JVCG	44.40	1.81%	12.41	45.44%
1000	4	ISCG	171.40	7.11%	16.04	60.41%
1000	4	IJCG	20.80	1.35%	25.84	48.12%
1000	8	MTCG	505.00	-	-	45.89%
1000	8	JVCG	64.20	2.33%	7.91	39.83%
1000	8	ISCG	452.00	-0.42%	1.12	46.51%
1000	8	IJCG	43.40	2.05%	12.25	40.13%
1500	4	MTCG	1448.60	-	-	48.49%
1500	4	JVCG	144.20	0.23%	10.14	40.82%
1500	4	ISCG	48.00	12.22%	30.67	56.02%
1500	4	IJCG	52.40	0.20%	29.19	41.95%
1500	8	MTCG	1656.40	-	-	57.95%
1500	8	JVCG	230.80	0.85%	7.25	51.41%
1500	8	ISCG	683.40	2.55%	7.15	61.07%
1500	8	IJCG	109.40	0.69%	15.59	51.54%
2000	4	MTCG	3433.60	-	-	53.02%
2000	4	JVCG	266.80	2.80%	13.03	43.74%
2000	4	ISCG	71.80	12.60%	48.89	59.66%
2000	4	IJCG	92.00	2.03%	39.27	46.19%
2000	8	MTCG	3434.80	-	-	48.43%
2000	8	JVCG	417.00	0.30%	8.26	43.56%
2000	8	ISCG	1510.00	4.61%	4.00	47.09%
2000	8	IJCG	206.40	0.13%	17.21	43.82%
2500	4	MTCG	6934.67	-	-	50.33%
2500	4	JVCG	549.00	0.26%	12.68	43.54%
2500	4	ISCG	129.67	9.70%	53.71	58.56%
2500	4	IJCG	139.33	1.78%	51.90	46.71%
2500	8	MTCG	5902.60	-	-	36.92%
2500	8	JVCG	672.60	-0.22%	8.72	31.51%
2500	8	ISCG	731.20	8.58%	13.42	35.96%
2500	8	IJCG	351.40	-0.61%	16.82	32.13%

Table 11 – Results with two simultaneous breaks

	Gap	SpeedUp	Route Changes
JVCG	2.22%	8.69	26.91%
ISCG	4.06%	11.72	32.42%
IJCG	1.93%	17.77	27.27%

The results considering two simultaneous disruptions at around 6 a.m. can be seen in table 11. Algorithm IJCG, in general, offer better solution quality (some with negative gaps) very efficiently. IJCG is the better trade-off algorithm because the gap is, in average, around 2% and the speedup factor almost 18 times faster than benchmark. Algorithm ISCG was the most disturbed by the breaks demonstrated by high gap, low speedup factor and the biggest percentage route exchanges.

7 Conclusions

In this paper, we presented a framework to solve the MDVTRSP in a fast approach. This problem is important, as disruptions happen in the daily schedules of every company, and the order of transportation should be restored as soon as possible. Such a problem requires a real-time solution because the results must be processed by operators and communicated to the bus drivers, which also takes time. As the size of the model is too big to be solved in such short time, we proposed two-phase fast heuristic algorithms to produce results in a couple of seconds. Our tests on randomly generated instances showed that the framework gives an excellent trade-off solution. Because of their ability to produce multiple good quality solutions in a short time, these algorithms seem suitable for a decision support system that helps the operators of a transportation company in the rescheduling process by giving them a possible solution for the problem.

To speed up the solution of very large instances, we developed a state space reduction procedure that selects, from the whole feasible solution space, a good set of arcs and uses it as an initial solution. With the use of these preprocessing procedures, the MDVTRSP can be solved with efficiency and efficacy. To the best of our knowledge, no other research has been successful in solving this problem in this manner.

The framework allows managers, after they detect schedule disruptions, to generate a really fast rescheduling. A column generation framework with state space reduction strategy was adopted. From computational experiments performed on randomly generated data, it is possible to observe that the algorithm that combines state space reduction and an initial solution obtained, on average, the best solutions, requiring the smallest CPU times, overcoming all remaining algorithms for the set of analyzed instances. So IJCG was the best trade-off procedure and this algorithm is our recommendation.

Future research is directed towards using the heuristic framework to develop novel approaches for solving the integrated crew and vehicle rescheduling problem in transportation contexts where the time and heterogeneous fleet is a relevant characteristic.

Final Remarks

This disseratation presented a three articles compilation in urban bus transportation optimization. The main objective was to study and implement heuristic solutions method based on Operations Research to optimizing offline and online vehicle (re)scheduling problems considering multiple depots and heterogeneous fleet.

In the first paper, a fast heuristic approach to deal with the multiple depot vehicle scheduling problem was proposed. First, the heuristics employ a combination of procedures to select, from the whole feasible solution space, a good set of arcs to compose the solution of the problem. Each procedure is based on the following selection criteria: (i) the set of paths found in the solution of $|K|$-SDVSPs; and (ii) the set of paths found in the solution of a relaxed MDVSP formulation, where the sequence of trips carried out by a vehicle can start and finish in different depots. The selected set of arcs is then solved by using a modified truncated CG algorithm. The developed approach led to very competitive method to solve the MDVSP.

Different combinations of the selection procedures were tested, resulting in three different variants. On the one hand, variant R1 achieved narrow gaps to best-known solutions (0.62% on average) with exceptional running times (roughly 34 times faster, on average, than previously reported results). On the other hand, the variant R1+R2 offered the best trade-off between solution quality and running times. It was, on average, 16 times faster than previously reported results Pepin et al. (2009), Otsuki e Aihara (2014), obtaining solutions with an average gap of around 0.5%. We think the main contributions are the column generation framework for large instances and the state-space reduction techniques for accelerating the solutions.

In the second paper, we added complexity when considering the heterogeneous fleet, denoted as "the multiple-depot vehicle-type scheduling problem" (MDVTSP). Although the MDVTSP importance and applicability, mathematical formulations and solution methods for it are still relatively unexplored. We introduced a new heuristic framework, combining time-space network, truncated column generation (TCG) and state space reduction, to solve large instances of the MDVTSP. With the use of these preprocessing procedures, the MDVTSP can be solved with efficiency and efficacy in comparison with a more traditional CG implementation.

From computational experiments performed on randomly generated data, it was possible to observe that the algorithm that combines state space reduction and an initial solution obtained, on average, the best solutions, requiring the smallest CPU times,

overcoming all remaining algorithms for the set of analyzed instances. This implementation is more efficient than the state-of-the-art in MDVTSP solution, considering the same number of depots and trips. However, in a real case instance, the best solution was obtained by the heuristic that uses only the state space reduction (SSR). Comparing with the current manual scheduling used by the transport consortium, savings of around 31% and 35% were obtained in total costs and in the number of vehicles, respectively. We think the main contribution is the column generation framework for instances with heterogeneous fleet since no other proposal in the literature has been identified at moment by the authors.

In the third part of this dissertation, however, we focused on the real-time schedule recovery for the case of serious vehicle failures. Such vehicle breakdowns require that the remaining passengers from the disabled vehicle, and those expected to become part of the trip, to be picked up. In addition, since the disabled vehicle may have future trips assigned to it, the given schedule may be deteriorated to the extent where the fleet plan may need to be adjusted in real-time depending on the current state of what is certainly a dynamic system. Usually, without the help of a rescheduling algorithm, the dispatcher either cancels the trips that are initially scheduled to be implemented by the disabled vehicle (when there are upcoming future trips planned that could soon serve the expected demand for the canceled trips), or simply dispatches an available vehicle from a depot. In both cases, there may be considerable delays introduced. This manual approach may result in a poor solution. The implementation of new technologies (e.g., automatic vehicle locators, the global positioning system, geographical information systems, and wireless communication) in public transit systems makes it possible to implement real-time vehicle rescheduling algorithms at low cost.

The framework allows managers, after they detect schedule disruptions, to generate a really fast rescheduling with, on average, around to 23 times faster than benchmarking at a cost of around 2% in objetive function value. If it happens two simultaneous breaks still around 18 times faster. The algorithm that combines state space reduction and an initial solution obtained, on average, the best solutions, requiring the smallest CPU times, overcoming all remaining algorithms for the set of analyzed instances. So IJCG was the best trade-off procedure and this algorithm is our recommendation.

The main contribution is the efficient approach to rescheduling under a disruption. The approach with integrated state-space reduction, initial solution, and column generation framework enable a really real-time action. In less than five minutes rescheduling all trips remaining.

Bibliography

BARNHART, C.; JOHNSON, E. L.; NEMHAUSER, G. L.; SAVELSBERGH, M. W. P.; VANCE, P. H. Branch-and-price: column generation for solving huge integer programs. Operations Research, v. 46, n. 3, p. 316–329, 1998.

BEASLEY, J. E.; CHRISTOFIDES, N. An algorithm for the resource constrained shortest-path problem. Networks, v. 19, n. 4, p. 379–394, 1989.

BERTOSSI, A. A.; CARRARESI, P.; GALLO, G. On some matching problems arising in vehicle scheduling models. Networks, v. 17, n. 3, p. 271–281, 1987.

BERTSEKAS, D. A simple and fast label correcting algorithm for shortest paths. Networks, v. 23, p. 703–709, 1993.

CACCHIANI, V.; HUISMAN, D.; KIDD, M.; KROON, L.; TOTH, P.; VEELENTURF, L.; WAGENAAR, J. An overview of recovery models and algorithms for real-time railway rescheduling. Transportation Research Part B: Methodological, v. 63, p. 15 – 37, 2014. ISSN 0191-2615. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0191261514000198>.

CAROSI, S.; GUALANDI, S.; MALUCELLI, F.; TRESOLDI, E. Delay management in public transportation: Service regularity issues and crew re-scheduling. Transportation Research Procedia, v. 10, p. 483 – 492, 2015. ISSN 2352-1465. 18th Euro Working Group on Transportation, EWGT 2015, 14-16 July 2015, Delft, The Netherlands. Disponível em: <http://www.sciencedirect.com/science/article/pii/S2352146515001891>.

CARPANETO, G.; DELL'AMICO, M.; FISCHETTI, M.; TOTH, P. A branch and bound algorithm for the multiple depot vehicle scheduling problem. Networks, v. 19, n. 5, p. 531–548, 1989.

CEDER, A. A. Optimal multi-vehicle type transit timetabling and vehicle scheduling. Procedia - Social and Behavioral Sciences, v. 20, n. 0, p. 19 – 30, 2011.

CEDER, A. A. Public-transport vehicle scheduling with multi vehicle type. Transportation Research Part C: Emerging Technologies, v. 19, n. 3, p. 485 – 497, 2011.

CLAUSEN, J.; LARSEN, A.; LARSEN, J.; REZANOVA, N. J. Disruption management in the airline industry-concepts, models and methods. Computers & Operations Research, v. 37, n. 5, p. 809 – 821, 2010. Disruption Management.

DÁVID, B.; KRÉSZ, M. A model and fast heuristics for the multiple depot bus rescheduling problem. In: 10th International Conference on the Practice and Theory of Automated Timetabling (PATAT). [S.l.: s.n.], 2014.

DELL'AMICO, M.; TOTH, P. Algorithms and codes for dense assignment problems: the state of the art. Discrete Applied Mathematics, v. 100, n. 1-2, p. 17–48, 2000.

DESAULNIERS, G.; HICKMAN, M. D. Public transit. In: BARNHART, C.; LAPORTE, B. (Ed.). Handbooks in Operations Research and Management Science, Transportation. [S.l.]: North-Holland, 2007. p. 69–127.

DOROGOVTSEV, S. N.; MENDES, J. F. F. Evolution of networks. Advanced Physics, v. 51, p. 1079 – 1187, 2002.

GINTNER, V.; KLIEWER, N.; SUHL, L. Solving large multiple-depot multiple-vehicle-type bus scheduling problems in practice. OR Spectrum, v. 27, p. 507–523, 2005.

GUEDES, P. C.; BORENSTEIN, D. Column generation based heuristic framework for the multiple-depot vehicle type scheduling problem. Computers & Industrial Engineering, Pergamon, v. 90, p. 361–370, 2015.

GUEDES, P. C.; LOPES, W. P.; ROHDE, L. R.; BORENSTEIN, D. Simple and efficient heuristic approach for the multiple-depot vehicle scheduling problem. Optimization Letters, v. 10, n. 7, p. 1449–1461, October 2015.

HADJAR, A.; MARCOTTE, O.; SOUMIS, F. A branch-and-cut algorithm for the multiple depot vehicle scheduling problem. Operations Research, v. 54, n. 1, p. 130–149, 2006.

HANE, C. A.; BARNHART, C.; JOHNSON, E. L.; MARSTEN, R. E.; NEMHAUSER, G. L.; SIGISMONDI, G. The fleet assignment problem: solving a large-scale integer program. Math. Program., Springer-Verlag New York, Inc., Secaucus, NJ, USA, v. 70, n. 2, p. 211–232, out. 1995.

HASSOLD, S.; CEDER, A. A. Public transport vehicle scheduling featuring multiple vehicle types. Transportation Research Part B: Methodological, v. 67, n. 0, p. 129 – 143, 2014. ISSN 0191-2615.

HUISMAN, D.; FRELING, R.; WAGELMANS, A. P. M. A robust solution approach to the dynamic vehicle scheduling problem. Transportation Science, v. 38, n. 4, p. 447–458, 2004.

HUISMAN, D.; FRELING, R.; WAGELMANS, A. P. M. Multiple-depot integrated vehicle and crew scheduling. Transportation Science, v. 39, n. 4, p. 491–502, 2005.

IBARRA-ROJAS, O.; DELGADO, F.; GIESEN, R.; MUÑOZ, J. Planning, operation, and control of bus transport systems: A literature review. Transportation Research Part B: Methodological, v. 77, p. 38–75, 2015.

JIN, J. G.; TEO, K. M.; ODONI, A. R. Optimizing bus bridging services in response to disruptions of urban transit rail networks. Transportation Science, INFORMS, v. 50, n. 3, p. 790–804, 2015.

JONKER, R.; VOLGENANT, A. A shortest augmenting path algorithm for dense and sparse linear assignment problems. Computing, Springer-Verlag, v. 38, n. 4, p. 325–340, 1987.

KLIEWER, N.; MELLOULI, T.; SUHL, L. A time-space network based exact optimization model for multi-depot bus scheduling. European Journal of Operational Research, v. 175, n. 3, p. 1616–1627, 2006.

LAURENT, B.; HAO, J.-K. Iterated local search for the multiple depot vehicle scheduling problem. Computers & Industrial Engineering, v. 57, n. 1, p. 277–286, 2009.

LI, J. Q.; HEAD, K. L. Sustainability provisions in the bus scheduling problem. Transportation Research Part D-Transport and Environment, v. 14, n. 1, p. 50–60, 2009.

LI, J.-Q.; MIRCHANDANI, P. B.; BORENSTEIN, D. Vehicle rescheduling problem: model and algorithms. Networks, v. 50, n. 3, p. 211–229, 2007.

LI, J.-Q.; MIRCHANDANI, P. B.; BORENSTEIN, D. A vehicle rescheduling problem with real-time vehicle reassignments and trip cancellations. Transportation Research Part E-Logistics and Transportation Review, v. 45, n. 3, p. 419–433, 2009.

Löbel, A. Vehicle scheduling in public transit and Lagrangean pricing. Management Science, v. 44, n. 12, p. 1637–1649, 1998.

LÜBBECKE, M. E.; DESROSIERS, J. Selected topics in column generation. Operations Research, v. 53, n. 6, p. 1007–1023, 2006.

MEHLHORN, K.; ZIEGELMANN, M. Resource constrained shortest paths. In: PATERSON, M. (Ed.). Algorithms - ESA 2000. [S.l.]: Springer Berlin Heidelberg, 2000, (Lecture Notes in Computer Science, v. 1879). p. 326–337.

OTSUKI, T.; AIHARA, K. New variable depth local search for multiple depot vehicle scheduling problems. Journal of Heuristics, p. 1–19, 2014.

OUGHALIME, A.; ISMAIL, W. R.; LIONG, C.-Y.; AYOB, M. B. Vehicle and driver scheduling modelling: A case study in ukm. In: Proceedings of the 2nd Conference on Data Mining and Optimization, DMO 2009, Universiti Kebangsaan Malaysia, 27-28 October 2009. [S.l.]: IEEE, 2009. p. 53–59.

OUKIL, A.; Ben Amor, H.; DESROSIERS, J.; El Gueddari, H. Stabilized column generation for highly degenerate multiple-depot vehicle scheduling problems. Computers and Operations Research, v. 34, p. 817–834, 2007.

PAIXÃO, J. M.; BRANCO, I. A quasi-assignment algorithm for bus scheduling. Networks, v. 17, n. 3, p. 249–269, 1987.

PEPIN, A.; DESAULNIERS, G.; HERTZ, A.; HUISMAN, D. A comparison of five heuristics for the multiple depot vehicle scheduling problem. Journal of Scheduling, v. 12, n. 1, p. 17–30, 2009.

RAMOS, J. A.; REIS, L. P.; PEDROSA, D. Solving heterogeneous fleet multiple depot vehicle scheduling problem as an asymmetric traveling salesman problem. In: EPIA. [S.l.: s.n.], 2011. p. 98–109.

RIBEIRO, C. C.; SOUMIS, F. A column generation approach to the multiple-depot vehicle scheduling problem. Operations Research, v. 42, n. 1, p. 41–52, 1994.

ROMAN, A. Top 100 bus fleet. Metro Magazine, p. 25–30, 2012.

SPLIET, R.; GABOR, A. F.; DEKKER, R. The vehicle rescheduling problem. Computers & Operations Research, Elsevier, v. 43, p. 129–136, 2014.

STEINZEN, I.; GINTNER, V.; SUHL, L.; KLIEWER, N. A time-space network approach for the integrated vehicle- and crew-scheduling problem with multiple depots. Transportation Science, v. 44, n. 3, p. 367–382, 2010.

Sun Tran. Sun Tran Monthly Operations Report, June 2005.

UÇAR, E.; BIRBIL, Ş. İ.; MUTER, İ. Managing disruptions in the multi-depot vehicle scheduling problem. Transportation Research Part B: Methodological, Elsevier, 2016.

van den HEUVEL, A. P. R. .; van den AKKER, J. M.; NIEKERK, M. E. van K. Integrating timetabling and vehicle scheduling in public bus transportation. The Netherlands, 2008. 17 p.

VANDERBECK, F. Implementing mixed integer column generation. In: DESAULNIERS, G.; DESROSIERS, J.; SALOMON, M. (Ed.). Column generation. [S.l.]: Springer, 2005. p. 331–358.

VISENTINI, M. S.; BORENSTEIN, D.; LI, J.-Q.; MIRCHANDANI, P. B. Review of real-time vehicle schedule recovery methods in transportation services. Jornal of Scheduling, v. 17, n. 6, p. 541–567, Dec 2014.

WOLSEY, L. A. Integer Programming. Chichester, USA: Wiley, 1998.

YILDIZ, E. Multi-depot vehicle scheduling with disruptions. Dissertação (Mestrado) — Sabanci University, Istanbul, Turkey, 2011.