OR Spectrum (2005) 27: 507–523
DOI: 10.1007/s00291-005-0207-9

OR Spectrum
© Springer-Verlag 2005

Solving large multiple-depot multiple-vehicle-type bus scheduling problems in practice

Vitali Gintner, Natalia Kliewer, and Leena Suhl

Decision Support & Operations Research Lab and International Graduate School for Dynamic Intelligent Systems, University of Paderborn, Warburger Straße 100, 33100 Paderborn, Germany (e-mail: {gintner, kliewer, suhl}@uni-paderborn.de)

Abstract. We consider the multiple-depot multiple-vehicle-type scheduling problem (MDVSP) which arises in public transport bus companies and aims to assign buses to cover a given set of timetabled trips with consideration of practical requirements, such as multiple depots and vehicle types as well as depot capacities. An optimal schedule is characterized by minimal fleet size and minimal operational costs including costs for empty movements and waiting time. It is well-known that the MDVSP is NP-hard.

Although progress has recently been made in solving large practical MDVSP to optimality with time-space network models, current optimization technology sets limits to the model size that can be solved. In order to approach very large practical instances we propose a two-phase method which produces close to optimal solutions. This modeling approach enables us to solve real-world problem instances with thousands of scheduled trips by direct application of standard optimization software. Furthermore, we introduce the concept of depot groups for the case that a bus may return in the evening into another depot than where it started in the morning.

Keywords: Multiple-depot vehicle scheduling – Public transport – Time-space network – Heuristic

1 The multiple-depot multiple-vehicle-type bus scheduling problem

We consider the scheduling of vehicles under constraints and objectives arising in urban and suburban public transport. Thus, each timetabled trip can be served by a vehicle (bus) belonging to a given set of vehicle types. There is a limited number of

Correspondence to: L. Suhl

buses available, each belonging to a given vehicle type, such as standard, double-decker, kneel bus etc. Each vehicle has to start and end its work day in one of the given depots. We understand a depot as a parking garage with limited space for buses to stay overnight.

After serving one timetabled trip, also called *service trip* or *scheduled trip*, each bus can serve one of the trips starting later from the station where the vehicle is standing, or it can change its location by moving unloaded to an another station, through an *empty movement*, in order to serve the next loaded trip starting there.

Our goal is to determine a set of routes for buses, called *vehicle schedule*, so that the total cost of serving a given timetable is minimized. The cost components include fixed costs for each vehicle as well as variable operational costs. The variable costs consist of distance-dependent travel costs and time-dependent costs for time spent outside the depot - the case where a driver is obliged to stay with the bus. All cost components depend on vehicle type. Since the fixed vehicle cost components are usually orders of magnitude higher than the operational costs, the optimal solution always involves the minimal number of vehicles.

The combinatorial complexity of this multi-depot bus scheduling problem (MDVSP[1] in the following) is determined by numerous possibilities to assign a vehicle type to each trip, to build sequences of trips for particular buses, and to assign buses to certain depots. To represent such sequences of trips, most exact modeling approaches known in the literature consider explicitly all possible connections – pairs of trips that can be served successively.

In the context of this paper, we refer to all models that represent connections explicitly as “connection-based” models. Because connection-based MDVSP models inherently include a very large number of arcs, such models of practical dimension with thousands of trips and multiple depots can often not be solved with standard optimization software packages. Known publications describe application of techniques such as column generation and Lagrangean relaxation for the multi-commodity flow formulation (see Löbel, 1999; Mesquita and Paixão, 1999), a general branch-and-bound approach Carpaneto et al. (1989) or an auction algorithm for the quasi- assignment formulation (see Freling, 1997) of the MDVSP. The combined vehicle and crew scheduling problem is discussed for the single depot problem in Friberg and Haase (1999), Haase et al. (2001), Freling (1997), and Freling et al. (2003), and for the multiple depot case in Huisman et al. (2003), as an example.

In cases where suboptimal solutions are sufficient, other authors propose heuristic methods, such as schedule first – cluster second approaches (Daduna and Mojsilovic, 1988; Daduna and Paixão, 1995) or shortest path – based heuristics (Dell’Amico et al., 1993).

Another approximate approach to cope with real-world instances is the artificial limitation of degrees of freedom for vehicle type and depot assigning. Furthermore, one may consider only a subset of possible empty trips (see, e.g., Löbel, 1999). Traditionally, mathematical optimization models for public mass transit use

[1] MDVSP means in a sense of this paper the MDMVTBSP – the multi-depot, multi-vehicle-type bus scheduling problem

connection-based networks where trips and depots are represented by nodes, and possible connections between trips given by arcs (see, e.g., Daduna and Paixão, 1995). The rotation of a vehicle is thus generated as a flow of one unit starting in a depot and returning back into it. The nodes connected by a flow represent the scheduled trips, carried out by the corresponding vehicle.

From our point of view, the explicit modeling of all possible connections seems to be inefficient for large scale problems, because the number of such pairs grows quadratically with the number of scheduled trips. Thus, our approach is based on an other underlying network structure, the so called *time-space network* which has been used since a long time in airline applications (Hane et al., 1995), but only recently in bus scheduling applications (Kliewer et al., 2002). In this paper, we show how to combine the exact time-space network modeling approach with heuristic components in order to be able to solve very large problems and take various practical requirements into account. Furthermore, we present how the solution time and quality can be improved if we ignore the requirement that a bus has to return to the same depot where it starts a route (one-depot-per-route requirement).

2 A time-space network based model formulation

We consider the bus scheduling problem where D denotes the set of depots and T the set of all possible vehicle types. The single-depot case ($|D| = 1$) can be modeled as a minimum cost flow problem and solved with efficient algorithms. Trips are represented by nodes, and vehicles represent the flow units circulating in the network. Each arc is provided with the cost of its corresponding activity - such as standing time cost, empty movement or fixed vehicle cost. The objective is to minimize the total flow costs over all arcs. All flow units involved in the single-depot problem are of the same type, so that the incoming and outgoing flow of each trip-node has to be equal to one. The mathematical model consists of the minimum cost flow problem with flow conservation constraints for each network node. No integrality constraints are needed because of the unimodularity of the coefficient matrix of the linear programming relaxation.

For the multi-depot case ($|D| > 1$), the network has to be multiplied so that there is a network layer for each depot. The mathematical formulation of this model contains additional restrictions in form of partitioning (covering) constraints, guaranteeing that each service trip is carried out by exactly one vehicle belonging to one of the required vehicle types. Because of the explicit integrality constraints (all flow variables must be equal to 0 or 1), the optimization model is no more a minimum cost flow model and becomes NP-hard (see Bertossi et al., 1987).

Furthermore, we need to model different bus types analogously to depots: For each bus/trip assignment we have to distinguish, from which depot the bus comes as well as which to which bus type it belongs. Thus, we do not only multiply the underlying network for each depot, but for each depot/bus type combination, which produces a network with $|D| \times |T|$ layers.

We propose a time-space network model where service trips are represented with two nodes: one for the departure and one for the arrival event, each event referring to the corresponding station. The network is complemented with possible

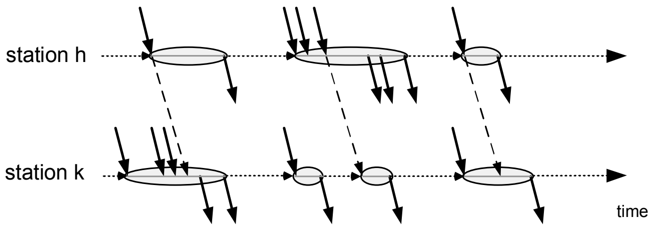

Fig. 1. Basic structure of the time-space network model

pull-out and pull-in trips from or to a depot, as well as all possible empty movements between two stations. Each station and depot is modeled with a time line connecting the departure and arriving events sorted according to time.

One special requirement in bus traffic is that empty movements are basically possible between all stations, and large savings can be achieved through an intelligent use of empty movements. This means, that after each service trip a bus may move to any of the other stations to take over a service trip starting elsewhere. However, it usually makes no sense to model such movements explicitly, because of the high combinatorial complexity. Through a node aggregation scheme presented in Kliewer et al. (2002) we were able to reduce the underlying network size often into a fraction of the original full size, still considering all possible empty movements. Our approach in this paper is based on this so-called *latest-first-matches* approach.

Figure 1 shows schematically the structure of a time-space network model for two stations h and k. At station h, for example, the arcs coming from above the time line represent incoming trips, and the arcs leaving down correspond to outgoing trips. If the problem contains m stations and n trips, the number of arcs in the aggregated time-space network is $O(nm)$ in contrast to $O(n^2)$ in the traditional connection-based approach, with $m << n$ for MDVSP-Instances (Kliewer et al., 2002).

One node of this network connects a group of possible arrivals to the following group of possible departures. In this manner all stations, including depots, are represented as ordered sets of (connection) nodes, linked by waiting arcs. In each network layer corresponding to a given depot/bus-type combination, there is one circulation flow arc connecting the last node (in time) of the depot line to its first node. This arc is provided with fixed cost for corresponding vehicle type and represents vehicles parking over night in depot. Depot capacities can thus be considered in different ways:

– We set upper bounds on circulation flow arcs to represent capacities of depot/vehicle-type combinations.
– Limitation of the number of parking slots in certain depots implies additional restrictions in the mathematical model formulation that constrain the sum of flow values on corresponding circulation flow arcs.
– For a limited number of vehicles of given type we constrain the sum of corresponding flow value as well.
– If it is required to use a fixed number of buses of certain type or of certain depot, we set a lower bound equal to the upper bound (variable fixing in the mathematical model) on the corresponding arc.

Operational costs of waiting arcs and empty movements can thus be modeled as costs of flow unit on the corresponding arcs. Upper bounds on the service-trip arcs are set equal to one.

The objective function basically consists of minimizing the total cost consisting of fixed cost for each vehicle used as well as variable operational cost for all arcs involved, so that costs of all arcs including the service arcs depend on vehicle type. Usually we would like to minimize the number of vehicles as the first objective, and then search for a cost-minimal solution among all vehicle-minimal solutions. This implies that we should choose the cost factors carefully: The costs on a circulation arc have to be large enough in comparison to the operational costs. Below we present cases where the cost coefficients have been set properly taking this requirement into account. .

More formally, the multi-layer network is constructed in the following way.

Mathematical formulation

Let $N = \{1, 2, \ldots, n\}$ be the set of trips, and let D be the set of depots (in the following, we define the depot as a combination of a depot and a vehicle type). We defined the vehicle scheduling network $G^d = (V^d, A^d)$ corresponding to depot d, which is an acyclic directed graph described above with nodes V^d and arcs A^d.

Let c^d_{ij} be the vehicle cost of arc $(i, j) \in A^d$, which is usually some function of travel and idle time. The vehicle cost of arcs representing idle time activity in the depot is 0. Furthermore, a fixed cost for using a vehicle is set on the circulation arc. Let $N^d(n) \in A^d$ be the arc corresponding to the trip n in the vehicle scheduling network G^d.

Decision variable x^d_{ij} indicates whether an arc (i, j) is used and assigned to the depot d or not. For each decision variable an upper bound is defined as follows:

$$u^d_{ij} = \begin{cases} 1\ , & \text{if } x^d_{ij} \text{ corresponds to a timetable trips} \\ u^d\ , & \text{if } x^d_{ij} \text{ corresponds to a circulation arc,} \\ & \quad \text{where } u^d \text{ is the capacity for depot } d \\ M\ , & \text{otherwise,} \\ & \quad \text{where } M \text{ is the maximum number of available vehicles} \end{cases}$$

The MDVSP can be formulated as follows.

$$min \quad \sum_{d \in D} \sum_{(i,j) \in A^d} c^d_{ij} x^d_{ij} \tag{1}$$

$$\sum_{\{j:(i,j) \in A^d\}} x^d_{ij} - \sum_{\{j:(j,i) \in A^d\}} x^d_{ji} = 0 \qquad \forall\, i \in V^d, \forall\, d \in D \tag{2}$$

$$\sum_{d \in D, (i,j) \in N^d(n)} x^d_{ij} = 1 \qquad \forall\, n \in N \tag{3}$$

$$0 \le x^d_{ij} \le u^d_{ij} \qquad \forall\, (i,j) \in A^d, \forall\, d \in D \tag{4}$$

$$x^d_{ij}\ integer \qquad \forall\, (i,j) \in A^d, \forall\, d \in D \tag{5}$$

The objective (1) is to minimize the sum of total vehicle costs. Constraints (2) are the typical flow conservation constraints, indicating that the flow into each node equals the flow out of each node, while constraints (3) assure that each trip must be covered by a vehicle. In this way we obtain a time-space network based multi-commodity flow formulation.

Computational results

Due to the aggregation of possible connections in the time-space network formulation, any feasible flow - especially also an optimal flow - represents *a bundle* or *a class* of vehicle schedules. All schedules within a given class have minimal total costs but different distributions of waiting times. With the help of a suitable flow decomposition procedure, we may extract a vehicle schedule with an optimal flow and desired characteristics. There are different ways ranging from a FIFO to a LIFO strategy to decompose a given optimal flow in a time-space network.

Thus, we solve a given MDVSP instance in following steps:

- Read the data and construct the network layers of a time-space network.
- Set up a mixed-integer optimization model for min cost flow with an additional cover constraint for each service trip.
- Optimize the MIP and compute an optimal flow representing a class of aggregated schedules.
- Decompose this flow to rotations of one special optimal vehicle schedule, according to LIFO, FIFO or any intermediate strategy.

In the following, we present numerical results for five real-life test problems which are described in Table 1:

- city1: a schedule of a regional bus company with three different bus types and 21 stations,
- city2: a schedule of a midsize German city with one bus type and 48 stations,
- city3: a schedule of a big German city with one bus type and 124 stations,
- mun1: instance of the city of Munich
- mun2: instance of the city of Munich

Table 1. Five test cases

	# Trips	# Stations	# Vehicle types
city1	2047	21	3
city2	3054	48	1
city3	7068	124	1
mun1	11062	160	12
mun2	10710	140	12

In the test cases only part of the depots and/or bus types could be included into the models when using exact methods: we used three bus types and up to nine depots for city1, one bus type and up to eight depots for city2 as well as one bus type and up to four depots for city3. These models could be optimally solved with less than ten hours on state-of-the-art hardware and software (see Table 2).

Table 2 shows the number of depots together with the number of network layers (the number of the depot/vehicle type combinations allowed) included into the model. Thus, for city1 the number of network layers is three times the number of depots because we build a network layer for each depot/bus type combination. Since not all bus types are allowed for each trip, we ignore the incompatible trip/bus type combinations in the model. This means that the number of scheduled trips within a network layer may be a subset of the total number of trips. The column øG denotes the average number of possible depot/bus type combinations for a trip. Furthermore, Table 2 contains the size of the MIP coefficient matrix as well as the CPU time on Intel Pentium with 2.2 GHz and 2 GB main memory. The models were solved with ILOG CPLEX 8.0 ILOG (2004).

Table 2. Computational results with exact optimization

	#Depots	#Network layers	øG	Rows	×	cols	CPU (sec)
city1	1	3	2.5	8798	×	27599	8
	2	6	5	14997	×	53249	114
	4	12	10	29031	×	118768	1380
	7	21	17.5	50647	×	214110	9228
	9	27	22.5	65223	×	284539	29040
city2	1	1	1	9597	×	64231	11
	2	2	2	16060	×	129295	53
	4	4	4	29076	×	249775	676
	6	6	6	41370	×	365185	5332
	8	8	8	54096	×	485834	27720
city3	1	1	1	20575	×	248021	42
	2	2	2	34072	×	501903	289
	3	3	3	47574	×	720181	16026
	4	4	4	73866	×	1210542	32400
mun1	18	55	6.5	162858	×	1479848	39020
mun2	3	15	15	159659	×	2032883	>100000

The test case mun1 contains as many as 18 depots and 55 network layers, however, the value of øG is ”only” 6.5, implying that we are still able to cope with the size of the problem using available main memory. The test problem mun2 involves 3 depots and 15 network layers, the value of øG being 15. As shown in Table 2 we were able to solve mun1 up to optimality in roughly 11 hours, but it was not possible to solve mun2 within three days.

We would like to point out that models of this size with unrestricted empty movements could not be solved to optimality before the time-space network model with aggregation was introduced. The tests with data of the city of Munich showed an improvement of several per cent compared to the current practice, and consequently the approach has been deployed for the bus scheduling application in Munich.

3 A variable fixing heuristic

The computational results in the previous sections show that we may find an optimal solution for small and mid-size models, but solving large practical instances of the MDVSP still remains a challenge. One way to cope with very large models is to switch over to heuristic methods which may compute a good solution fast but cannot guarantee optimality of the solution found. In this section, we propose a basic heuristic that reduces the model size through fixing part of the original variables, so that it is possible to solve the reduced problem to optimality.

The basic idea of the heuristic is to first solve a number of simplified models, for example a single-depot vehicle scheduling problem for each depot, and then search for common chains of trips in each of the solutions. If the same sequence of trips is included in each solution, we denote it as a *stable chain* and assume that it may occur in the optimal global solution as well. The stable chains are then treated as trips in an exact optimization model, so that the model size can be significantly reduced. Because we first fix some variables and then use exact optimization techniques, we call this approach *fix-and-optimize*.

Figure 2 illustrates the outline of the fix-and-optimize approach. First, we decompose the given MDVSP into several simplified problems, usually consisting of one given depot in the geographical sense (and not in the sense of depot/bus type combinations, meaning that we consider several bus types as belonging to the depot). After solving these easier problems, we go through all solutions and determine stable chains as robust sequences of trips. After fixing them, the remaining smaller MDVSP can often be solved with standard optimization packages. Of course, it depends on data whether the resulting model can be solved to optimality, however, our tests with real-life models are very promising, as we will show later in this section.

Figure 3 outlines an algorithm to determine stable chains among M different sets of routes, each set corresponding to a single-depot VSP with a given depot/bus-type combination. Let T denote the set of all service trips, S_m the set of service trips within a trip chain m, and S_{all} the set of all trip chains that have been determined. Furthermore, for each set of routes m there exists an assignment function $F_m : T \to T$ such that for a given trip $t_i \in T$ the value $F_m(t_i)$ gives the successor of t_i within the chain to which both belong.

4 Computational results

This fix-and-optimize heuristic was tested with the real-life cases described in Section 2. It turned out that often a very significant improvement of the solution

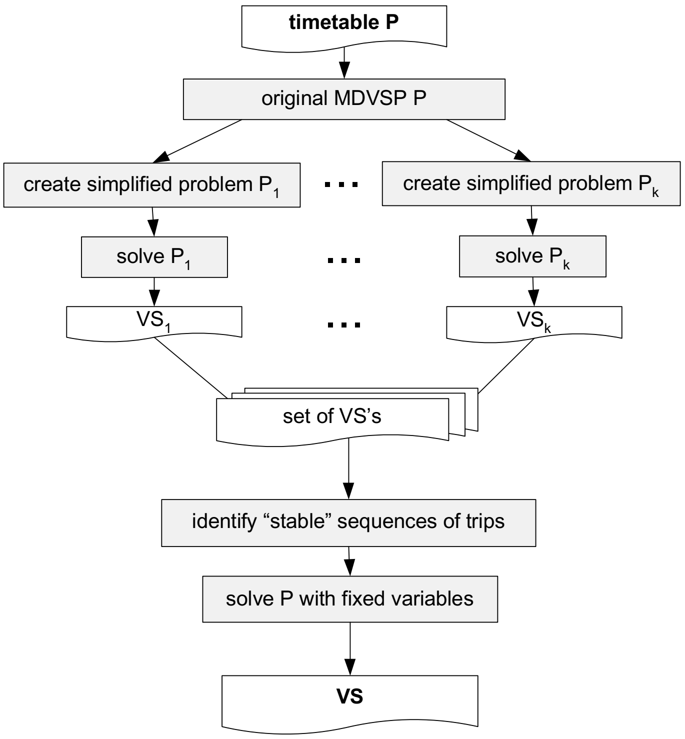
timetable P
original MDVSP P
create simplified problem P_1
create simplified problem P_k
solve P_1
solve P_k
VS_1
VS_k
set of VS's
identify "stable" sequences of trips
solve P with fixed variables
VS

Fig. 2. Outline of the heuristic fix-and-optimize approach

time was possible with very good solution quality. In those cases in which the optimal solution is known, we notice that the ratio between the heuristic and the exact solution is far below one per cent, more precisely between 0.07 and 0.25 % for the regional model, 0.01 and 0.10 % for the urban small model, and 0.001 and 0.0001 % for the urban large model. The results were computed with ILOG CPLEX 8.0 (ILOG, 2004). Except for the largest problem instances, we also received acceptable run times with the optimizer MOPS (MOPS, 2004).

Table 3 and Figure 4 compare the results of exact optimization with the fix-and-optimize approach for city1. The line "#depots" shows the number of depots, varying between 2 and 9, together with the number of network layers in brackets, which varies between 6 and 27, correspondingly. With growing number of network layers, the number of unfixed trips grows implying that more freedom is available, and the problem becomes more difficult. We conclude that the quality of the heuristic solution is very high, whereas the running time of the heuristic is only between 15 and 30 % of the exact solution time. As Figure 4 demonstrates, the difference in total running time gets more significant when the problem size increases.

The behavior of fix-and-optimize for the test problem city2 is illustrated in Table 4 and Figure 5. In this case only one bus type is involved, meaning that the number of depot/bus type combinations is identical to the number of depots and varies between 2 and 8. Even in the case with eight depots, about 2/3 of the trips can be fixed, implying very fast solution times of the approximate optimization

```
Procedure Algorithm to determine stable chains
$S = \emptyset$
$S_{all} = \emptyset$

for $i = 1$ to $N - 1$
    $S = S \cup t_i$
    ChainEndFlag = 0
    for $m = 2$ to $M$
        if $F_m(t_i)$ <> $F_1(t_i)$ then
            ChainEndFlag = 1
            break for
        endif
    endfor
    if ChainEndFlag then
        $S_{all} = S_{all} \cup S$
        $S = \emptyset$
    endif
endfor
$S_{all} = S_{all} \cup F_N$
```

Fig. 3. A basic algorithm to determine stable chains

Table 3. Computational experiments with fix-and-optimize: city1

#Depots	2(6)	3(9)	4(12)	5(15)	6(18)	7(21)	8(24)	9(27)
#Unfixed trips	967	1135	1232	1390	1408	1482	1498	1538
% from 2047	47.2	55.4	60.2	67.9	68.8	72.4	73.2	75.1
Operational cost								
Exact	1027.7	935.4	925.9	789.0	747.8	685.1	647.2	643.3
Heuristic	1028.5	937.8	928.2	789.7	748.3	686.3	648.6	644.1
GAP(%)	0.07	0.25	0.25	0.09	0.07	0.19	0.22	0.14
CPU (min)								
Exact	1.9	10.9	23.0	56.6	142.8	153.0	234.0	484.0
Heuristic	0.5	1.5	6.9	8.2	27.5	54.8	75.8	118.0

problem. Indeed, the solution time for the exact model after fix-and-optimize is only roughly between one and two per cent of the time needed to solve the original model.

The test model city3 shows a similar behavior as city1 and city2, as illustrated in Table 5 and Figure 6. In the case of five depots, the original model could not be solved within three days, but it was possible to solve the reduced model after fix-and-optimize in roughly two hours. Thus, from a practical point of view, the heuristic fixing procedure may enable us to find nearly optimal solutions for large-scale problems that cannot be approached with exact solution techniques today. It

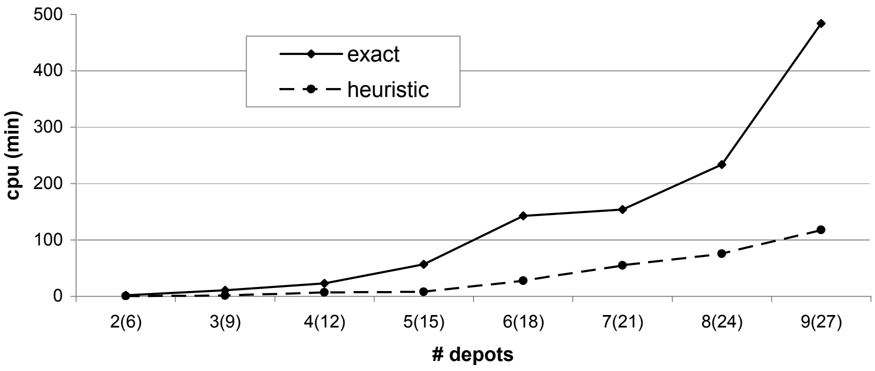

Fig. 4. Running time versus number of depots: city1

Table 4. Computational experiments with fix-and-optimize: city2

#Depots	2	3	4	5	6	7	8
#Unfixed trips	745	891	939	939	971	992	1004
% from 3054	24.4	29.2	30.7	30.7	31.8	32.5	32.9
Operational cost							
Exact	55424.8	54705.8	54486.2	54486.2	54301.4	54325.1	54138.1
Heuristic	54428.6	54752.2	54535.1	54535.1	54357.7	54351.9	54184.6
GAP(%)	0.01	0.08	0.09	0.09	0.10	0.05	0.09
CPU (min)							
Exact	0.9	6.2	11.3	22.3	88.7	370.0	462.0
Heuristic	0.4	0.7	1.3	2.2	3.6	5.7	6.9

turned out that the solution quality reached with fix-and-optimize is significantly better than the quality when using manual scheduling as is the case currently in practice.

In order to cope with difficult instances of the test cases muc1 and muc2, we first built simplified approximate models through aggregation of depots and bus types. For muc1 we constructed two simplified time-space network flow oriented models, each with 52 depots. Each simplified model produced a bundle of optimal solutions which was decomposed into a vehicle schedule according to four different criteria. Thus, we produced two, four, or eight different vehicle schedules under which stable chains were identified, as shown in Table 6. As the line "#unfixed trips" shows, most of the trips could be fixed, so that the time to solve the reduced problem became almost negligible. The line "heuristic" under "CPU (min)" first denotes the CPU time to solve both simplified models (for example 301 min. for 2 vehicle schedules), and second the (almost negligible) time to solve the remaining approximate exact optimization model (for example 1.5 min. for 2 vehicle schedules).

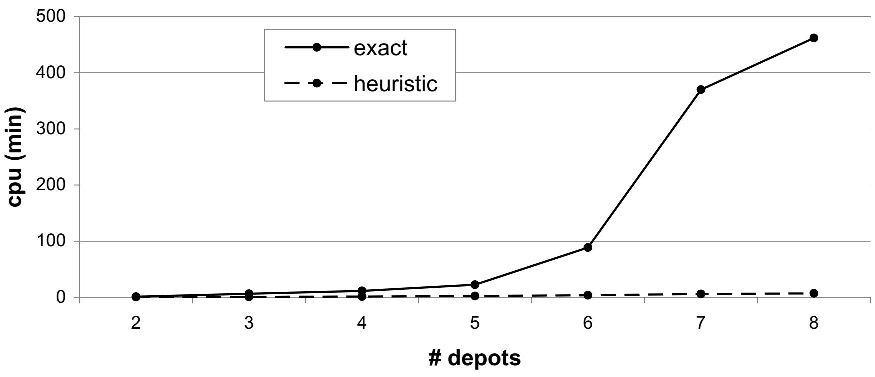

Fig. 5. Running time versus number of depots: city2

Table 5. Computational experiments with fix-and-optimize: city3

#Depots	2	3	4	5
#Unfixed trips	2269	2443	2630	2653
% from 7068	32.1	34.6	37.2	37.5
Operational cost				
Exact	75922.0	75480.0	74930.3	–
Heuristic	75928.8	75481.8	74931.4	74839.2
GAP(%)	0.01	0.003	0.001	–
CPU (min)				
Exact	4.9	267.1	540.0	–
Heuristic	2.4	7.8	31.4	74.6

For mun2, we solved a simplified case with three depots (instead of the original 15), modifying the capacity restrictions respectively. We decomposed the optimal solutions according to two strategies, resulting in two different vehicle schedules. The approximate model was solved in only 41 seconds, and only 1646 trips out of 10710 remained unfixed. The resulting approximate optimization problem was then solved within 11 minutes. In other words, the fix-and-optimize heuristic was able to compute a solution in a total time of roughly 12 minutes (11 minutes plus 41 seconds), whereas the original model could not be solved within 100,000 minutes to optimality. Because the optimal solution is not known, no precise statement on the heuristic solution quality is possible.

To summarize, we conclude that the fix-and-optimize approach is a good choice for very large MDVSP, because it significantly increases the size of problems for which close to optimal solutions can be found.

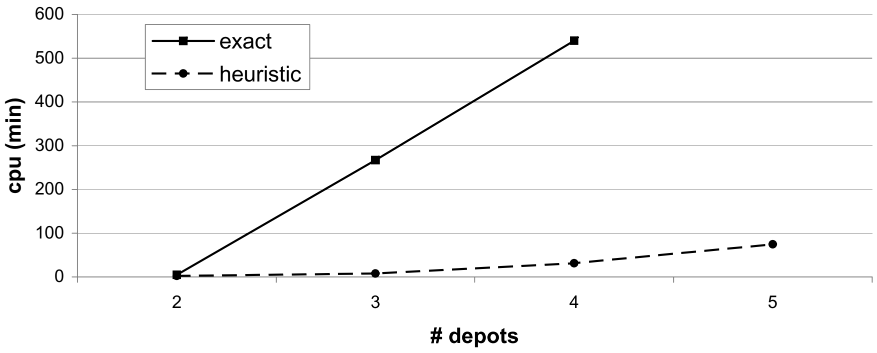

Fig. 6. Running time versus number of depots: city3

Table 6. Computational experiments with fix-and-optimize: mun1

#Considered vehicle schedules	2	4	8
#Unfixed trips	1438	2396	2528
% from 11062	13.0	21.7	22.9
Operational cost			
Exact	17,831,010	17,831,010	17,831,010
Heuristic	17,946,839	17,831,739	17,831,739
GAP(%)	0.65	0.004	0.004
CPU (min)			
Exact	651	651	651
Heuristic	301 + 1.5	301 + 1.8	301 + 2

5 The concept of depot groups

The basic MDVSP model presented above requires that at the end of a day each bus returns to the same depot where it started in the morning. Furthermore, it assumes that the driver takes all breaks in this depot. This is advantageous, because the time for a break at the home depot is not paid in contrary to break time spent outside the depot.

However, dropping these requirements fully or partially may imply significant savings in many practical situations. It may be advantageous to have a break at another depot or bus stop than the home depot of a driver. This is the case if the pull-in trip to the home depot would be long compared to the trip to another depot closer to the bus route. Some bus stops, for example close to a railway station or a shopping center, are suitable for breaks as well. In general, if the last trip of a day is closer to another depot than the first trip, it may be advantageous that a daily route ends at another depot than where it starts.

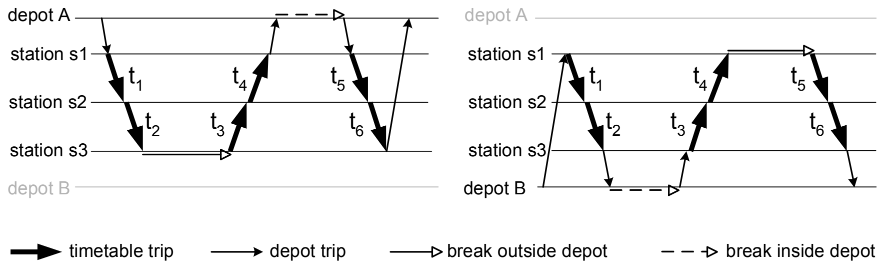

Fig. 7. An example with one-depot-per-tour

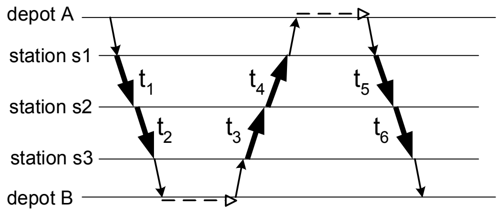

Fig. 8. An example with different start and end depots per tour

The example in Figures 7 and 8 illustrates the situation where costs can be saved in allowing a bus to use two different depots. The example involves six trips t_1 to t_6, two depots A and B, and one bus. The bus starts in the morning at station $s1$ close to depot A and ends after the tour at station $s3$ close to depot B. Figure 7 shows the routes if we allow the bus to use only either depot A or depot B. In both cases there is one long pull-in or pull-out trip (from $s3$ to A for depot A, and B to $s1$ for depot B). Furthermore, it is not possible to have a cost-efficient break at a depot, but the driver has to take the break at station $s3$ (when using depot A) or station $s1$ (when using depot B), both of which have to be paid as working time.

Figure 8 shows the case that the bus in our example leaves from depot A in the morning and stays at depot B at the end of the tour. In this case both the pull-out and the pull-in trips are short, and the driver can have both breaks at a depot without extra cost.

For practical reasons it is usually not desired to allow a completely free depot selection. For example, the driver often has to finish close to the original depot where he or she started, so that it is convenient to get home at the end of the working day. Thus, depots that are fairly close to each other may be grouped together into a depot group. Furthermore, there might be other logical reasons for grouping several depots together. This is the case when during the morning and afternoon many trips pass by schools, and in the evening many trips pass by free time activities, such as sports or music facilities. Considering depot groups instead of single depots reduces the problem size, and may thus provide a meaningful way to construct a model which is solvable with today's technologies.

In the following, we modify the multi-layer optimization model (1)–(5) in order to represent depot groups instead of single depots. This means that the set K represents depot-group/bus-type pairs instead of depot/bus-type combinations. Thus

$K = G \times T$, where G is the set of depot groups and T the set of bus types. We use the variable g for depot groups, where $g \in G$.

Because a given depot may belong to several depot groups, we have to modify the constraints expressing overnight parking capacity: It is not enough to record the backward arc per depot group, but we need an individual backward arc for each depot which may belong to several groups. Let $x_{dd}^{g,t}$ denote the variable for the backward arc corresponding to depot d as part of group g. The total number of buses standing overnight in depot d is computed as a sum over all depot groups to which the depot d belongs (6).

$$\sum_{g \in G with d \in g} x_{dd}^{g,t}, \forall\, d \in D, \forall\, t \in T \tag{6}$$

In the following, we present computational results for the test problems city1, city2, and city3 when using two or three depots.

Table 7 demonstrates the results for the two-depot-case, and Table 8 for the three-depot-case. In both cases we used three bus types for the city1, and one bus type for city2 and city3, resulting in six respectively nine network layers for city1 and one network layer for city2 and city3. The computational environment was as above.

Table 7. Computational results for the two-depot-models

	#Network layers	Rows	×	cols	Costs	cpu (sec)
city1						
single-depot-tour	6	16812	×	60409	1027.7	190.3
with depot groups	3	10593	×	33314	1025.2	9.1
city2						
single-depot-tour	2	16064	×	128001	55424.8	47.3
with depot groups	1	10438	×	68081	55419.3	10.7
city3						
single-depot-tour	2	34076	×	490719	75921.7	278.7
with depot groups	1	21758	×	241876	75917.9	46.7

The results in Tables 7 and 8 show that the depot-group based models can be solved much faster than the original models with the singe-depot-tour concept. Furthermore, the total cost is lower with the depot group concept. Especially, the best solution with three depot groups is significantly better than the best solution with two depot groups: for the regional model it is 9.3%, for the urban small model 1.45% and the urban large model 1.11% better. It is generally true that we achieve better solutions through the use of depot groups, because the solution space is larger compared to the single-depot-route-concept.

Table 8. Computational results for the three-depot-models

	#Network layers	Rows	×	cols	Costs	cpu (sec)
city1						
single-depot-tour	9	23314	×	92105	935.4	695.7
with depot groups	3	11938	×	37769	929.8	11.4
city2						
single-depot-tour	3	22257	×	188015	54705.8	271.6
with depot groups	1	11028	×	68032	54614.3	10.5
city3						
single-depot-tour	3	47194	×	722825	75095.1	10783.0
with depot groups	1	22570	×	225167	75073.5	42.7

6 Conclusions

The multiple-depot multiple-vehicle-type scheduling problem in urban bus transit is one of the most difficult optimization problems in traffic and transportation, and generally cannot be solved optimally with state-of-the-art technologies. In this paper, we presented suboptimal methods and relaxations in order to be able to find close to optimal solutions for problems of practical size in reasonable time.

We proposed a fix-and-optimize method based on the time-space network approach for the multiple-depot multiple-vehicle-type scheduling problem. We first build several simplified models with one or two depots but including all trips and, after solving the simplified models to optimality, identify chains of trips that are carried out with one and the same vehicle in each solution. We fix such chains, so that it becomes in many cases possible to solve the corresponding smaller model with all original depots to optimality. Practical case studies at regional and urban bus companies demonstrate that it is possible to solve far larger problems close to optimality than is the case in using the pure MIP approach.

Furthermore, we investigated the concept of depot groups where depots are divided into groups so that a bus may start and end at any depot of a given group. Together with the concept of allowing breaks outside a depot we were able to find better solutions in shorter time than the original one-depot-per-tour model.

We conclude that it is possible and favorable to solve large practical MDVSP with an exact mixed-integer optimization approach combined with heuristic variable fixing. Furthermore, various practical requirements can be taken into account, so that the savings induced by the approach of this paper can be realized in practical applications.

References

Bertossi, A.A., Carraresi, P., Gallo, G. (1987) On some matching problems arising in vehicle scheduling models. Networks 17: 271–281

Carpaneto, G., Dell'Amico, M., Fischetti, M., Toth, P. (1989) A branch and bound algorithm for the multiple depot vehicle scheduling problem. Networks 19: 531–548

Daduna, J.R., Mojsilovic, M. (1988) Computer-aided vehicle and duty scheduling using HOT programme system. In: Daduna, J.R., Wren, A. (eds.) Computer-aided transit scheduling, pp. 133–146. Springer, Berlin Heidelberg New York

Daduna, J.R., Paixão, J.M.P. (1995) Vehicle scheduling for public mass transit – an overview. In: Daduna, J.R., Branco, I., Paixão, J.M. Pinto (eds.) Computer-aided transit scheduling, pp. 76–90. Springer, Berlin Heidelberg New York

Dell'Amico, M., Fischetti, M., Toth, P. (1993) Heuristic algorithms for the multiple depot vehicle scheduling problems. Management Science 39: 115–125

Forbes, M.A., Hotts, J.N., Watts, A.M. (1994) An exact algorithm for multiple depot vehicle scheduling. European Journal of Operational Research 72: 115–124

Freling, R. (1997) Models and techniques for integrating vehicle and crew scheduling. Ph.D. thesis, Tinbergen Institute, Erasmus University Rotterdam

Freling, R., Huisman, D., Wagelmans, A.P.M. (2003) Models and algorithms for integration of vehicle and crew scheduling. Journal of Scheduling 6: 59–81

Friberg, C., Haase, K. (1999) An exact branch and cut algorithm for the vehicle and crew scheduling problem. In: Wilson, N.H.M. (ed.) Computer-aided transit scheduling, pp. 63–80. Springer, Berlin Heidelberg New York

Haase, K., Desaulniers, G., Desrosiers, J. (2001) Simultaneous vehicle and crew scheduling in urban mass transit systems. Transportation Science 35: 286–303

Hane, C., Barnhart, C., Johnson, E.L., Marsten, R.E., Nemhauser, G.L., Sigismondi, G. (1995) The fleet assignment problem: solving a large integer program. Mathematical Programming 70: 211–232

Huisman, D., Freling, R., Wagelmans, A.P.M. (2003) Multiple-depot integrated vehicle and crew scheduling. Tech. report EI2001-17, Econometric Institute, Erasmus University Rotterdam

ILOG (2004) www.ilog.com, December 30, 2004

Kliewer, N., Mellouli, T., Suhl, L. (2002) A new solution model for multi-depot multi-vehicle-type vehicle scheduling in suburban public transport. In: Proceedings of the 13th Mini-EURO Conference and the 9th Meeting of the EURO Woring Group on Transportation, Bari, Italy

Löbel, A. (1999) Solving large-scale multiple-depot vehicle scheduling problems. In: Wilson, N.H.M. (ed.) Computer-aided transit scheduling, pp. 193–220. Springer, Berlin Heidelberg New York

Mesquita, M., Paixão, J. (1999) Exact algorithms for the multiple-depot vehicle scheduling problem based on multicommodity network flow type formulations. In: Wilson, N.H.M. (ed.) Computer-aided transit scheduling, pp. 223–246. Springer, Berlin Heielberg New York

MOPS (2004) www.mops-optimize.com, December 30, 2004

Ribeiro, C.C., Soumis, F. (1994) A column generation approach to the multiple-depot vehicle scheduling problem. Operations Research 42: 41–52