# Vehicle and Crew Scheduling: Solving Large Real-World Instances with an Integrated Approach

Sebastiaan W. de Groot[1] and Dennis Huisman[2]

[1] ORTEC bv, Gouda, the Netherlands sgroot@ortec.nl
[2] Erasmus Center for Optimization in Public Transport (ECOPT) & Econometric Institute, Erasmus University Rotterdam, P.O. Box 1738, NL-3000 DR Rotterdam, the Netherlands huisman@few.eur.nl

**Summary.** In this paper we discuss several methods to solve large real-world instances of the vehicle and crew scheduling problem. Although there has been an increased attention to integrated approaches for solving such problems in the literature, currently only small or medium-sized instances can be solved by such approaches. Therefore, large instances should be split into several smaller ones, which can be solved by an integrated approach, or the sequential approach, i.e., first vehicle scheduling and afterwards crew scheduling, is applied.

In this paper we compare both approaches, where we consider different ways of splitting an instance varying from very simple rules to more sophisticated ones. Those ways are extensively tested by computational experiments on real-world data provided by the largest Dutch bus company.

## 1 Introduction

In the literature on vehicle and crew scheduling, not much attention has been paid to the problem of splitting up large instances into several smaller ones such that a good overall solution is obtained. Algorithms are developed to solve a certain problem, either optimally or heuristically, and they are tested on self made problem instances, or on (small) instances from practice which the algorithm can still solve. If a real-world instance has to be solved and it seems to be too large for the algorithm to solve it, the problem is just split up into several smaller instances, the algorithm is used to solve those smaller instances and the results are combined such that there is an overall solution. This solution is then feasible, but of course, even if the algorithm itself provides an optimal solution, optimality for the overall problem is likely to be lost. The way the instance has been divided up is almost never an issue in the literature. However, different divisions can result in completely different final outcomes; one splitting can result in a much better solution than another one. Therefore, the instances are mostly divided according to some logical rules.

For example, in the field of crew scheduling, Fores *et al.* (2001) describe this problem. In 1998, they subdivided a large instance of ScotRail into two smaller instances according to a geographic division. Since this resulted in some strange outcomes, several tasks were exchanged between the different divisions. After several days of trial and error, they found a reasonable splitting of the instance such that the optimal solutions of both smaller instances seemed to give a reasonable overall solution. In 2000, they were able to solve the large instance optimally. They checked the performance of the splitting and indeed the optimal solution of the complete instance was the same as the solution which they obtained by splitting up the instance several years before.

Haghani *et al.* (2003) describe a comparative analysis of different approaches to solve large-scale vehicle scheduling problems with route time constraints. This can be seen as a special case of the integrated vehicle and crew scheduling problem, namely where a duty exactly coincides with a vehicle and the only constraint is a maximum duty length. They compared several approaches on a large real-world instance in Baltimore which consists of multiple depots. Since they could not solve this problem exactly, they considered three approaches. The first approach (see also Haghani and Banihashemi (2002)) used CPLEX to solve a reduced problem instance, i.e., several variables in the large IP were just omitted. In the second and third approach, they solved several smaller, single-depot instances with an exact algorithm. The difference between both approaches is the way in which the problem is split up. One is based on the current solution of the public transport company, the other on the outcome of the first approach. They showed that this last approach outperformed the first one.

For the integrated vehicle and crew scheduling problem only small and medium-sized instances have been solved (see, e.g., Huisman *et al.* (2005)). Therefore, we try to answer the following questions in this paper.

1. How can large instances be split up into several smaller ones such that applying an integrated approach on those instances can be done in a reasonable computation time?
2. Does such a splitting approach outperform the sequential approach when the latter is used to solve the large instance at once?
3. Does it outperform the integrated approach when this is terminated after a certain computation time?

Furthermore, we compare different ways of splitting the problem and we give some results on several real-world instances from Connexxion. Finally, we use these ideas to find a solution for large problem instances which we could not solve before with an integrated approach.

The paper is organized as follows. In Section 2, we describe the integrated vehicle and crew scheduling problem and summarize a mathematical formulation and algorithm for this problem, which we introduced in an earlier paper (Huisman *et al.* (2005)). We discuss several splitting approaches in Section 3. Finally, a computational study is provided in Section 4.

## 2 Multiple-Depot Integrated Vehicle and Crew Scheduling

Several approaches to tackle the integrated variant of the vehicle and crew scheduling problem are recently proposed in the literature (see, e.g., Freling (1997), Haase and Friberg (1999), Haase *et al.* (2001) and Freling *et al.* (2003) for the single-depot case, and Gaffi and Nonato (1999), Huisman *et al.* (2005) and Huisman (2004) for the multiple-depot case). In Huisman *et al.* (2005), two different algorithms are proposed. Both are based on different mathematical formulations, which are themselves extensions of the single-depot case formulations proposed by Freling *et al.* (2003) and Haase *et al.* (2001), respectively. Because the first algorithm performed slightly better, we will only consider this one in the remainder of the paper. Before we discuss that algorithm, we will first provide a formal problem definition and a mathematical formulation.

### 2.1 Problem Definition

The *multiple-depot vehicle and crew scheduling problem* (MD-VCSP) combines the *multiple-depot vehicle scheduling problem* (MDVSP) and the *crew scheduling problem* (CSP). Given a set of *trips* within a fixed planning horizon, it minimizes the total sum of vehicle and crew costs such that both the vehicle and the crew schedule are feasible and mutually compatible. Each trip has fixed starting and ending times, and can be assigned to a vehicle and a crew member from a certain set of depots. Furthermore, the travelling times between all pairs of locations are known. A vehicle schedule is feasible if (1) all trips are assigned to exactly one vehicle, and (2) each trip is assigned to a vehicle from a depot that is allowed to drive this trip. From a vehicle schedule it follows which trips have to be performed by the same vehicle and this defines so-called vehicle *blocks*. The blocks are subdivided at *relief points*, defined by location and time, where and when a change of driver may occur and drivers can enjoy their break. A *task* is defined by two consecutive relief points and represents the minimum portion of work that can be assigned to a crew. These tasks have to be assigned to crew members. The tasks that are assigned to the same crew member define a crew *duty*. Together the duties constitute a crew schedule. Such a schedule is feasible if (1) each task is assigned to one duty, and (2) each duty is a sequence of tasks that can be performed by a single crew, both from a physical and a legal point of view. In particular, each duty must satisfy several complicating constraints corresponding to work load regulations for crews. Typical examples of such constraints are maximum working time without a break, minimum break duration, maximum total working time, and maximum duration. Finally, a *piece (of work)* is defined as a sequence of tasks on one vehicle block without a break that can be performed by a single crew member without interruption.

We distinguish between two types of tasks, viz., *trip tasks* corresponding to trips, and *dh-tasks* corresponding to deadheading. A *deadhead* is a period that a vehicle is moving to or from the depot, or a period between two trips that a vehicle is outside of the depot (possibly moving without passengers).

## 2.2 Mathematical Formulation

Let $N = \{1, 2, ..., n\}$ be the set of trips, numbered according to increasing starting time. Define $D$ as the set of depots and let $s^d$ and $t^d$ both represent depot $d$. Moreover, define $E$ as the set of *compatible trips*, where two trips $i$ and $j$ are compatible if a vehicle can perform trip $j$ directly after trip $i$. We define the vehicle scheduling network $G^d = (V^d, A^d)$, which is an acyclic directed network with nodes $V^d = N^d \cup \{s^d, t^d\}$, and arcs $A^d = E^d \cup (s^d \times N^d) \cup (N^d \times t^d)$. Note that $N^d$ and $E^d$ are the parts of $N$ and $E$ corresponding to depot $d$, since it is not necessary that all trips can be served from every depot. Let $c^d_{ij}$ be the vehicle cost of arc $(i, j) \in A^d$.

To reduce the number of constraints, we assume that a vehicle returns to the depot if it has an idle time between two consecutive trips which is long enough to let it return. In that case the arc between the trips is called a *long arc*; the other arcs between trips are called *short arcs*. Denote $A^{sd}$ $(A^{ld})$ as the set of short (long) arcs.

Furthermore, $K^d$ denotes the set of duties corresponding to depot $d$ and $f^d_k$ denote the crew cost of duty $k \in K^d$, respectively. The subset of duties covering the trip task corresponding to trip $i \in N^d$ is denoted by $K^d(i)$, where we assume that a trip corresponds to exactly one task. $K^d(i, j)$, $K^d(s^d, j)$ and $K^d(i, t^d)$ denote the set of duties covering dh-tasks corresponding to deadhead $(i, j)$, $(s^d, j)$ and $(i, t^d) \in A^d$, respectively. Decision variables $y^d_{ij}$ indicate whether an arc $(i, j)$ is used and assigned to depot $d$ or not, while $x^d_k$ indicates whether duty $k$ corresponding to depot $d$ is selected in the solution or not. The MD-VCSP can then be formulated as follows.

$$\min \sum_{d \in D} \sum_{(i,j) \in A^d} c^d_{ij} y^d_{ij} + \sum_{d \in D} \sum_{k \in K^d} f^d_k x^d_k \tag{1}$$

$$\sum_{d \in D} \sum_{j:(i,j) \in A^d} y^d_{ij} = 1 \quad \forall i \in N \tag{2}$$

$$\sum_{d \in D} \sum_{i:(i,j) \in A^d} y^d_{ij} = 1 \quad \forall j \in N \tag{3}$$

$$\sum_{i:(i,j) \in A^d} y^d_{ij} - \sum_{i:(j,i) \in A^d} y^d_{ji} = 0 \quad \forall d \in D, \forall j \in N^d \tag{4}$$

$$\sum_{k \in K^d(i)} x^d_k - \sum_{j:(i,j) \in A^d} y^d_{ij} = 0 \quad \forall d \in D, \forall i \in N^d \tag{5}$$

$$\sum_{k \in K^d(i,j)} x^d_k - y^d_{ij} = 0 \quad \forall d \in D, \forall (i,j) \in A^{sd} \tag{6}$$

$$\sum_{k \in K^d(i,t^d)} x^d_k - y^d_{it^d} - \sum_{j:(i,j) \in A^{ld}} y^d_{ij} = 0 \quad \forall d \in D, \forall i \in N^d \tag{7}$$

$$\sum_{k \in K^d(s^d,j)} x^d_k - y^d_{s^d j} - \sum_{i:(i,j) \in A^{ld}} y^d_{ij} = 0 \quad \forall d \in D, \forall j \in N^d \tag{8}$$

$$x^d_k, y^d_{ij} \in \{0, 1\} \quad \forall d \in D, \forall k \in K^d, \forall (i,j) \in A^d \tag{9}$$

The objective is to minimize the sum of total vehicle and crew costs. The first three sets of constraints, (2)-(4), correspond to the formulation of the MDVSP. Constraints (5) assure that each trip task will be covered by a duty from a depot if and only if the corresponding trip is assigned to this depot. Furthermore, constraints (6), (7) and (8) guarantee the link between dh-tasks and deadheads in the solution, where deadheads corresponding to short and long arcs in $A^d$ are considered separately.

## 2.3 Algorithm

An outline of the algorithm is shown in Fig. 1.

**Step 0**: Initialization
Solve MDVSP and CSP for every depot and take as initial set of columns the duties in the CSP-solution.
**Step 1**: Computation of dual multipliers
Solve a Lagrangian dual problem with the current set of columns. This gives a lower bound for the current set of columns.
**Step 2**: Deletion of columns
If there are more columns than a certain minimum amount, then delete columns with positive reduced cost greater than a certain threshold value.
**Step 3**: Generation of columns
Generate columns with negative reduced cost.
Compute an estimate of a lower bound for the overall problem. If the gap between this estimate and the lower bound found in Step 1 is small enough (or another termination criterion is satisfied), go to Step 4;
otherwise, return to Step 1.
**Step 4**: Construction of feasible solution
Solve a second Lagrangian dual problem with the set of columns generated in Step 3, where the optimal solution of the subproblem gives feasible vehicle schedules. Solve for each depot the crew scheduling problem corresponding to the feasible vehicle schedules.

**Fig. 1.** Solution Method for MD-VCSP

First, we compute a feasible solution by using the sequential approach, which means we compute the optimal solution of the MDVSP and afterwards, we solve for each depot a CSP given the vehicle schedule for that depot. To solve the MDVSP, we use the model described in Huisman *et al.* (2004) and the all-purpose solver CPLEX. The approach we used to solve the CSP is described in Freling *et al.* (2003).

The main part of the algorithm is used to compute a lower bound and we use therefore a column generation algorithm. The *master problem* is solved with Lagrangian Relaxation. Furthermore, we generate the duties in the column generation subproblem (*pricing problem*). For details about the master and pricing problem, we refer to Huisman *et al.* (2005). Since we do not want to get a very large master problem, columns with high positive reduced costs will be removed. This only happens if there are more columns than a certain minimum number. Finally, in Step 4 we compute feasible solutions.

## 3 Different Ways of Splitting

In this section we describe several approaches of splitting a large instance of the MD-VCSP into several smaller ones. The different approaches can be divided into two categories:

1. splitting the problem into several single-depot vehicle and crew scheduling problems (SD-VCSPs), i.e., assign each trip to a depot;
2. splitting an instance into a predetermined number of smaller ones.

We will start the discussion with the first category. The most simple way is a random assignment of the trips to the depots. Although this is not interesting in itself, a more sophisticated rule should always beat this trivial one. The more interesting assignments of trips to depots are the following:

- assign each trip to the depot closest to its start location;
- assign each trip to the depot closest to its end location;
- assign each trip to the depot closest to a combination of its start and end location;
- solve the MDVSP and assign each trip to the depot where it is assigned to in the MDVSP.

The first three rules are based on the geographical structure of the problem and can be based on distances or travel times. However, the last rule requires solving of another, much simpler, optimization problem, namely the multiple-depot vehicle scheduling problem, and uses that solution. Note that even the MDVSP is a $\mathcal{NP}$-hard problem. Moreover, recall that the solution approach on the MD-VCSP starts with solving the MDVSP to obtain an initial feasible solution. Therefore, the extra effort is very low. Of course, it is possible to recombine certain smaller SD-VCSPs again to larger MD-VCSPs. This is especially attractive if certain subproblems are so small that recombining does not result in a too large problem again. Another possibility is to use this assignment only as a splitting of the instance and to consider more depots again during the optimization.

The second category is dividing the trips instead of the depot(s) into several small subproblems. We assume here that we have given a maximum number of trips per subproblem. This leads to a certain minimum number of subproblems. Below, we give an overview of such divisions.

- Assign each trip arbitrarily to a subproblem such that the maximum number of trips in a subproblem is not exceeded.
- Solve the MDVSP and assign all trips executed by the same vehicle to the same subproblem. However, the vehicles themselves are assigned arbitrarily to a subproblem.
- Solve the MDVSP and assign all trips executed by the same vehicle to the same subproblem. Moreover, assign the vehicles in consecutive order to the subproblems.

- Solve the MDVSP and assign all trips executed by the same vehicle to the same subproblem. Moreover, assign the vehicles with the highest correlation to the same subproblem.

The first three ways of dividing speak for themselves. The fourth one needs some further explanation. We calculate the correlation $w_{ij}$ between two vehicle blocks with the algorithm suggested in Fig. 2.

```
$w_{ij} := 0$.
For each different line number $l$ in vehicle block $i$:
        $\delta_i :=$ number of trips in block $i$ with line number $l$;
        $\delta_j :=$ number of trips in block $j$ with line number $l$;
        if $\delta_j > 0$, then $w_{ij} := w_{ij} + \delta_i + \delta_j - 1$;
        otherwise, $w_{ij} := w_{ij}$.
```

**Fig. 2.** Algorithm to Compute $w_{ij}$

It can be easily seen that the weight is only positive if both vehicle blocks have at least one trip in common of the same bus line.

We define a weighted graph $G = (V, E)$ with $V$ as the set of nodes, where a node corresponds to a vehicle block and $E$ as the set of edges. There is an edge $(i, j)$ between each pair of nodes with its weight equal to $w_{ij}$. The assignment of the vehicle blocks to different subproblems corresponds now to the partitioning of the graph in certain subgraphs such that the total weight of the cuts is minimal and the different parts have an (almost) equal size, where the size of a part is defined as the sum of the number of trips executed by each vehicle block in that part. A well-known algorithm for bipartition is the one of Kernighan and Lin (1970). Hendrickson and Leland (1993) have generalized this algorithm for partitioning in more than two parts. We use this algorithm to partition our graph.

After the problem has been divided into several subproblems and they have been solved with an integrated approach, we can still recombine some parts of the problem such that the solution can be improved. Since the last step of the algorithm consists of solving a CSP for a certain vehicle schedule, we can recombine all vehicle schedules for each depot and solve one large CSP. Notice that this is possible, since the bottleneck of solving an integrated approach is not the CSP. We will see in the next section that this recombining significantly improves the solutions.

## 4 Computational Results

In this section we test our algorithms on two large data sets from Connexxion, which is the largest bus company in the Netherlands. The first set consists of 1104 trips and four depots in the area between Rotterdam, Utrecht and Dordrecht, three large cities in the Netherlands. The second set contains 1372 trips and six depots in the triangle

Rotterdam, Hoek van Holland, Leiden. We use eight subsets of the first set to test the splitting methods described in the previous section. Then, we choose the best one and perform that approach on the total set. This approach is also used to tackle the second set. The eight subsets are called instance 1 until 8, the complete set 1 is called instance 9 and set 2 is instance 10. In Subsection 4.1 we describe some other properties of these data instances.

All tests in this subsection are executed on a Pentium IV 1.8GHz personal computer (512MB RAM) with the following parameter settings. Notice that all computation times are denoted in minutes.

1. The objective is to minimize the total sum of vehicles and duties, i.e., we only consider fixed costs and the cost of a vehicle is equal to the cost of a duty. For solving the MDVSP in the sequential approach and in the initial step for the integrated approach we use an additional fictitious cost in the variable vehicle costs, viz., for every minute a vehicle is empty outside the depot a cost equal to 1 is incurred.
2. The pricing problems are solved independently for each depot and each type of duty. Moreover, we generate at most 1500 duties for each combination of a depot and type of duty.
3. The maximum number of iterations in the subgradient algorithm to solve the master problem (Step 1) is $500 + 3k$ in the $k$-th iteration of the column generation algorithm. However, for constructing the feasible solutions in Step 4, the number of iterations is only 10, since in that case the subproblem is $\mathcal{NP}$-hard. Such a small number of iterations is sufficient, since we already start with good multipliers, namely the best ones of the last iteration in the previous step. We construct 10 feasible solutions from which the best one will be selected.
4. The column generation algorithm is stopped if the difference between the current and estimated lower bound is smaller than $0.1\%$ or if the computation time of the lower bound phase is more than 4 hours (2 hours for cases where the problem is divided). Notice that in the latter case we do not have a proven lower bound.

## 4.1 Properties of the Real-World Data Instances

The restrictions that we have taken into account are as follows. A driver can only be relieved by another driver at the start or end of a trip at certain specified locations or at the depot. If a driver starts/ends his duty at the depot, there is a sign-on/sign-off time of 10 and 5 minutes, respectively. If a driver starts/ends his duty at another relief location, an extra time of 15 minutes plus the deadhead time between this location and the depot is added to the length of the duty. There are five different types of duties, one tripper type consisting of one piece with a length between 30 minutes and 5 hours, and four normal types consisting of two pieces with the properties described in Table 1.

**Table 1.** Properties of the Different Duty Types

| type | 1 (early) | | 2 (day) | | 3 (late) | | 4 (split) | |
|---|---|---|---|---|---|---|---|---|
| | min | max | min | max | min | max | min | max |
| start time | | | 8:00 | | 13:15 | | | |
| end time | | 16:30 | | 18:14 | | | | 19:30 |
| piece length | 0:30 | 5:00 | 0:30 | 5:00 | 0:30 | 5:00 | 0:30 | 5:00 |
| break length | 0:45 | | 0:45 | | 0:45 | | 1:30 | |
| duty length | | 9:45 | | 9:45 | | 9:45 | | 12:00 |
| work time | | 9:00 | | 9:00 | | 9:00 | | 9:00 |

## 4.2 Sequential and Integrated Approach

In Table 2, an overview of the results of the sequential and the integrated approach is provided. For each instance, we give the number of trips and the average number of depots to which a trip may be assigned. Furthermore, we give the number of vehicles, duties and the sum of these two as well as the computation time for the sequential and the integrated approach. Finally, we report the best lower bound given by the integrated approach. As can be seen from this table the integrated approach gives much better results than the sequential one. We were only able to compute lower bounds for five of the eight instances, given the maximum computation time of 4 hours for the lower bound phase.

**Table 2.** Results Without Splitting

| | instance | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| | number of trips | 194 | 210 | 220 | 237 | 304 | 386 | 451 | 653 |
| | av. depots/trip | 1.60 | 2.47 | 1.52 | 2.38 | 2.48 | 1.27 | 1.67 | 1.74 |
| seq. | vehicles | 19 | 33 | 27 | 34 | 40 | 32 | 47 | 67 |
| | duties | 35 | 56 | 49 | 62 | 75 | 61 | 86 | 125 |
| | V+D | 54 | 89 | 76 | 96 | 115 | 93 | 133 | 192 |
| | cpu (min.) | 1 | 0 | 0 | 0 | 1 | 2 | 2 | 3 |
| int. | vehicles | 19 | 33 | 27 | 34 | 40 | 32 | 47 | 67 |
| | duties | 29 | 52 | 40 | 55 | 66 | 59 | 75 | 117 |
| | V+D | 48 | 85 | 67 | 89 | 106 | 91 | 122 | 184 |
| | cpu (min.) | 155 | 32 | 94 | 43 | 244 | 260 | 254 | 275 |
| | lower | 44 | 77 | 64 | 81 | 95 | - | - | - |

## 4.3 Assigning Trips to Depots

In Section 3 we suggested four different methods to assign a trip to a depot. These approaches have been tested to split real-world Instance 2 (see Subsection 4.1), containing four depots, into two subproblems. Notice that this can be done in seven different ways (four with a single-depot and a 3-depot instance and three with two

2-depot instances). Table 3 provides the results of these divisions where the trips are assigned to a depot at random (average results over three runs), or using one of the four methods, i.e., closest to the start location, closest to the end location, closest to a combination of start and end location or according to the solution of the MDVSP. Notice that, e.g., 12-34 means that Depots 1 and 2 are in one subdivision, while 3 and 4 are in the other one.

**Table 3.** Sum of Vehicles and Crew Duties with Splitting Depots – Instance 2

| | 123-4 | 124-3 | 134-2 | 234-1 | 12-34 | 13-24 | 14-23 | av. |
|---|---|---|---|---|---|---|---|---|
| random | 95 | 99 | 93.7 | 93 | 91.7 | 101.7 | 95.3 | 95.6 |
| start | 104 | 104 | 89 | 88 | 89 | 110 | 102 | 98.0 |
| end | 96 | 101 | 90 | 86 | 91 | 101 | 97 | 94.6 |
| start-end | 94 | 98 | 90 | 83 | 88 | 99 | 92 | 92.0 |
| MDVSP | 86 | 87 | 85 | 83 | 84 | 87 | 86 | 85.4 |

From Table 3 we can immediately conclude that dividing based on the MDVSP is much better than on one of the geographical rules. Some of these do not even outperform a random assignment. We refer to De Groot (2003) for similar results on other instances. Therefore, we will only consider these types of divisions of the depots in the remainder of this section.

## 4.4 Splitting of the Trips

The different methods for the second category introduced in Section 3 have been tested on the eight real-world problem instances discussed in Subsection 4.1. We refer to De Groot (2003) for a detailed overview of the results of these tests. Here, we only provide an overview of those methods that performed well. These are the following methods.

- Solve the MDVSP and assign each trip to the depot where it is assigned to in the MDVSP. Afterwards divide the trips into two sets: one set with the trips assigned to the largest depot, i.e., the one with most trips assigned to it, and the other set with the remainder of trips. Divide those sets again into sets of at most 200 trips such that the trips executed by the same vehicle (resulting from the earlier solved MDVSP) should be in the same subproblem and the vehicles are assigned to the different subproblems in consecutive order (**Method A**).
- Same as Method A. However, the vehicles are now divided such that the ones with high correlation are as much as possible in the same subproblem (**Method B**).
- Same as Method A. However, the depots are not split first (**Method C**).
- Same as Method B. However, the depots are not split first (**Method D**).
- Same as Method C. However, the subproblems consists of at most 150 trips instead of 200 (**Method E**).

- Same as Method D. However, the subproblems consists of at most 150 trips instead of 200 (**Method F**).

Before we continue our discussion on methods of the second category, we first look at the effect of recombining the different crew scheduling problems per depot at the end. Since the effect on the computation time of this step can be neglected, we only compare the solution values. In Table 4 we provide this comparison for Method C.

**Table 4.** Sum of Vehicles and Crew Duties With/Without Recombining CSPs – Method C

| instance | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| with | 49 | 86 | 70 | 89 | 105 | 91 | 122 | 182 |
| without | 49 | 87 | 71 | 91 | 108 | 91 | 126 | 188 |

As can be seen from Table 4 the saving of recombining can be quite large (up to six duties). Therefore, we recommend to use this option always and thus we take this option into account for the other methods as well.

In Table 5, we report the total number of duties and the maximum computation time for one subproblem (cpu) in minutes for the methods A until F. The number of vehicles is not mentioned since it is independent of the method and the same as in Table 2. The total computation time is also not mentioned, since one of the advantages of splitting is that the algorithm can run on parallel machines.

**Table 5.** Results Splitting on Instances 1 - 8

| | instance | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| | trips | 194 | 210 | 220 | 237 | 304 | 386 | 451 | 653 |
| | depots/trip | 1.60 | 2.47 | 1.52 | 2.38 | 2.48 | 1.27 | 1.67 | 1.74 |
| A | duties | 31 | 51 | 43 | 57 | 66 | 59 | 75 | 117 |
| | cpu | 17 | 7 | 5 | 7 | 20 | 72 | 44 | 30 |
| B | duties | 31 | 51 | 43 | 57 | 66 | 58 | 77 | 117 |
| | cpu | 17 | 7 | 5 | 7 | 20 | 56 | 47 | 36 |
| C | duties | 29 | 53 | 43 | 55 | 65 | 59 | 75 | 115 |
| | cpu | 155 | 3 | 9 | 2 | 27 | 59 | 34 | 22 |
| D | duties | 29 | 53 | 43 | 56 | 66 | 58 | 74 | 114 |
| | cpu | 155 | 3 | 7 | 3 | 32 | 127 | 42 | 41 |
| E | duties | 30 | 53 | 43 | 55 | 67 | 57 | 75 | 118 |
| | cpu | 10 | 3 | 9 | 2 | 13 | 9 | 12 | 12 |
| F | duties | 31 | 53 | 43 | 56 | 66 | 58 | 76 | 118 |
| | cpu | 18 | 3 | 8 | 3 | 6 | 19 | 17 | 12 |

If we look at the results we need to make a distinction between Instance 1, Instances 2-5, Instance 6, and Instances 7 and 8. For Instance 1, Methods C and D provide the same results as the standard integrated approach, since there is no splitting at all. Furthermore, Methods A and B are the same. That is, the problem is divided into two subproblems, which reduces the computation time significantly but needs two duties more. For Instances 2-5 Methods A and B are the same. Here, we can see that the solutions are mostly slightly worse if we split the problems. However, the computation times reduce significantly. As mentioned earlier, for the largest three instances, the lower bound phase of the integrated approach was terminated after a maximum computation time and then feasible solutions were constructed. Here, we can already see an important benefit of the splitting idea. The solutions of some of the methods are better, while the others are equal. Moreover, the computation times are reduced dramatically. For the Instances 7 and 8, we can even see that most of the splitting methods provide better results. Moreover, the computation times become reasonably small. If we would run the subproblems on parallel machines the computation time would be less than one hour on each machine. For all instances, we can see that splitting the problem leads to much better results than the fast and simple sequential approach. If we compare the different methods with each other, we can conclude that Methods A and B perform worse than the others. If we compare C with D and E with F, i.e., using a more advanced approach to divide the vehicle blocks over the subproblems, then we can conclude that they are quite similar. Therefore, it does not make much sense to use this more complicated division. Moreover, if we compare E with C or F with D, then we see that the impact of smaller subproblems (at most 150 or 200 trips), is significant on the computation time, which could be expected of course, but small on the quality of the solutions. Altogether, we conclude that Method E performs well and has a low computation time. Therefore, we will use this one in the next subsection to solve the large instances.

## 4.5 Large Instances

Since we have shown that these methods to split an instance perform well, we consider the two large data sets introduced in the beginning of this section. Recall that those sets consist of 1104 and 1372 trips, and are called Instance 9 and 10, respectively. Furthermore, notice that the Instances 1 until 8 were derived from Instance 9 and that Instance 10 is completely independent. Although Instances 9 and 10 have four and six depots, on average each trip can only be assigned to 1.71 and 3.64 depots, respectively. Since Method E performed as the best one in the previous subsection, we use this method here. Moreover, we compared it with the sequential approach and the integrated approach with a maximum computation time. The results are shown in Table 6.

As can be seen from this table, the computation time of the integrated approach can far exceed the time limit of 4 hours for computing a lower bound. This can be explained by the fact that other steps take more time. For instance, the computation time of the MDVSP is about 9 and 35 minutes for Instances 9 and 10, respectively, while this was negligible before. Moreover, it can take some time before an iteration in the lower bound phase is finished. Since an iteration is always finished, the final computation time of the lower bound phase can exceed the time limit. Finally, the computation of the CSPs in Step 4 takes longer and this is done 10 times for

**Table 6.** Results Splitting on Instances 9 & 10

|  | instance | 9 | 10 |
|---|---|---|---|
|  | vehicles | 109 | 117 |
| seq | duties | 185 | 224 |
|  | cpu | 10 | 46 |
| int | duties | 179 | 219 |
|  | cpu | 336 | 474 |
| E | duties | 178 | 210 |
|  | cpu | 35 | 62 |

each subproblem. We can also see that the computation time of one subproblem in Method E can rise over one hour, while it was at most 13 minutes before. This can be explained by the larger sizes of the subproblems. Although the maximum size of a subproblem is 150 trips, this was never reached before. For these larger instances the number of trips in a subproblem comes closer to this maximum.

If we look at the results, we can see that the splitting method saves 7 and 14 duties compared to the sequential approach, and 1 and 9 duties compared to the integrated one. This is a reduction in labor force of 0.6% and 4.1%, respectively, which is quite significant. Moreover, the computation times are reduced drastically. Therefore, we can conclude that these splitting methods clearly outperform the sequential approach as well as the integrated one with a time limit.

## 5 Conclusions

In this paper we discussed several methods to split large problem instances of the integrated vehicle and crew scheduling problem into several smaller instances. We first applied these approaches to small instances, where we were able to calculate lower bounds on the optimal solutions and a feasible solution with the integrated approach on the complete instance. We showed that the effect of dividing these instances did not deteriorate the quality of the solutions a lot. Later on, we applied these ideas to large instances and showed that those could be solved now, which was not possible before. Furthermore, we showed that the saving compared with the simple, sequential approach is large. Finally, we recommend the use of such splitting methods to solve practical instances instead of dividing the problem in a 'logical' way.

## References

De Groot, S. W. (2003). Een geïntegreerde aanpak van voertuig- en personeelsplanning toegepast op grote probleeminstanties, master's thesis (in Dutch). School of Economics, Erasmus University Rotterdam.

Fores, S., Proll, L., and Wren, A. (2001). Experiences with a flexible driver scheduler. In S. Voß and J. R. Daduna, editors, *Computer-Aided Scheduling of Public Transport*, pages 137–152. Springer, Berlin.

Freling, R. (1997). *Models and Techniques for Integrating Vehicle and Crew Scheduling*. Ph.D. thesis, Tinbergen Institute, Erasmus University Rotterdam.

Freling, R., Huisman, D., and Wagelmans, A. P. M. (2003). Models and algorithms for integration of vehicle and crew scheduling. *Journal of Scheduling*, **6**, 63–85.

Gaffi, A. and Nonato, M. (1999). An integrated approach to extra-urban crew and vehicle scheduling. In N. H. M. Wilson, editor, *Computer-Aided Transit Scheduling*, pages 103–128. Springer, Berlin.

Haase, K. and Friberg, C. (1999). An exact branch and cut algorithm for the vehicle and crew scheduling problem. In N. H. M. Wilson, editor, *Computer-Aided Transit Scheduling*, pages 63–80. Springer, Berlin.

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

Haghani, A. and Banihashemi, M. (2002). Heuristic approaches for solving large-scale bus transit vehicle scheduling problem with route-time constraints. *Transportation Research Part A*, **36**, 309–333.

Haghani, A., Banihashemi, M., and Chiang, K.-H. (2003). A comparative analysis of bus transit vehicle scheduling models. *Transportation Research Part B*, **37**, 301–322.

Hendrickson, B. and Leland, R. (1993). An improved spectral load balancing method. In R. F. Sincovec, D. E. Keyes, M. R. Leuze, L. R. Petzold, and D. A. Reed, editors, *Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing*, pages 953–961. SIAM.

Huisman, D. (2004). *Integrated and Dynamic Vehicle and Crew Scheduling*. Ph.D. thesis, Tinbergen Institute, Erasmus University Rotterdam.

Huisman, D., Freling, R., and Wagelmans, A. P. M. (2004). A robust solution approach to the dynamic vehicle scheduling problem. *Transportation Science*, **38**, 447–458.

Huisman, D., Freling, R., and Wagelmans, A. P. M. (2005). Multiple-depot integrated vehicle and crew scheduling. *Transportation Science*, **39**, 491–502.

Kernighan, B. and Lin, S. (1970). An efficient heuristic procedure for partitioning graphs. *Bell Systems Technical Journal*, **29**, 291–307.