A Hybrid Evolutionary Algorithm for the Vehicle and Crew Scheduling Problem in Public Transit

Ingmar Steinzen, Matthias Becker and Leena Suhl

Abstract—The vehicle and crew scheduling problem in public transit aims at finding minimum cost bus and crew schedules such that all trips of a given timetable are operated respecting all operational constraints. In this paper we present a novel hybrid evolutionary algorithm for the multiple-depot integrated vehicle and crew scheduling problem that combines mathematical programming techniques with an evolutionary algorithm. Computational results on randomly generated benchmark instances demonstrate that our approach outperforms the traditional sequential treatment of vehicle and crew scheduling. Furthermore, it is competitive with solution approaches from literature that fully integrate both planning problems.

I. INTRODUCTION

Vehicle and crew scheduling play an important role in the management of a public transport company since these are the first planning steps where the primary focus is put on minimizing costs.

The integrated vehicle and crew scheduling problem (VCSP) for a given set of timetable trips within a fixed planning horizon and a fleet of vehicles assigned to several depots can be stated as follows: find a minimum cost vehicle and crew schedule such that both vehicle and crew schedule are feasible and mutually compatible. Vehicle and crew schedule are compatible if each vehicle activity in the vehicle schedule is also covered by exactly one duty while all activities not contained in the vehicle schedule are not part of any duty. Additionally, traveling times between all pairs of locations are known.

A vehicle schedule is feasible if each trip of a given timetable is assigned to exactly one vehicle and each vehicle performs a feasible sequence of trips. A sequence of trips (vehicle block) is feasible if each pair of consecutive trips can be executed in sequence and each block starts and ends at the same depot. Trips operated in sequence by the same vehicle are linked by deadheads. Deadheads are vehicle movements or idle times (or both) without carrying passengers. A vehicle is idle if it stands (idle) at a location other than the depot. A vehicle block is a sequence of compatible trips that starts with a pull-out trip and ends with a pull-in trip. A pull-out trip connects the depot with the start location of the first trip while a pull-in trip moves a vehicle from the end location of the last trip back to the depot. A daily schedule for one vehicle can thus include several vehicle blocks. The vehicle costs comprise of fixed (asset) costs for every vehicle and variable costs for travel and idle time outside the depot.

A crew schedule is feasible if each task (deadhead(s) and/or trip(s)) of the vehicle schedule is covered by a duty that can be performed by a single driver. Each duty must comply with a wide variety of federal laws, safety regulations, and collective in-house agreements, such as break rules, min/max working time, driving time or spread time (duty length). A task is a sequence of activities on a vehicle (like performing trips and/or deadheads) between two consecutive relief points and represents an elementary portion of work that can be assigned to a driver. A relief point defines a location and time where a driver may change his vehicle. A piece of work is a sequence of tasks without a (long) break for which a driver remains on the same vehicle. Consequently, duties are composed of pieces of work separated by breaks. The duty cost usually consists of a fixed component for wages and variable costs for working time or overtime bonuses.

If there are multiple depots some trips possibly have to be assigned to vehicles and drivers from a certain (subset) of depots. It is easy to see that a problem with multiple depots reduces to several single depot problems if every trip can only be serviced from a single depot. It is well known that the single depot vehicle scheduling problem corresponds to a minimum cost flow problem that can be solved in polynomial time while its multiple-depot counterpart is NP-hard. Furthermore, the crew scheduling problem is NP-hard with either working time or spread time constraints. Thus, the integrated vehicle and crew scheduling problem is NP-hard.

While partially integrated approaches for vehicle and crew scheduling problems have been proposed in literature until the early eighties (Ball et al. [1]), fully integrated approaches have lately attracted several researchers like Borndörfer et al. [5], Huisman et al. [9], or Mesquita and Paias [17] who all developed models and solution techniques mainly based on decomposition approaches for mathematical programming. Despite its success in solving a wide range of combinatorial optimization problems [18], evolutionary algorithms have not been applied to the integrated vehicle and crew scheduling problem. However, there are numerous successful applications of evolutionary algorithms for crew scheduling problems (e.g. Kwan et al. [13], Marchiori and Steenbeek [16], Li and Kwan [14]).

In this paper, we suggest a hybrid evolutionary algorithm that combines an evolutionary algorithm with different column generation methods to evaluate the fitness of individuals.

The paper is organized as follows. In Section II we give a mathematical formulation for the integrated vehicle and crew scheduling problem. We discuss a decomposition approach of...
the formulation in Section III. The decomposition approach provides the basis for the hybrid evolutionary algorithm that we describe in Section IV. Finally, we conclude the paper with some computational results on randomly generated instances in Section V.

II. MATHEMATICAL FORMULATION

In this section we discuss a mathematical formulation for the integrated vehicle and crew scheduling problem with multiple depots. The formulation was introduced by Gintner et al. [7] and combines a multicommodity network flow problem for vehicle scheduling with a set partitioning problem for crew scheduling. The main advantage of this formulation is the structure of the underlying vehicle scheduling network that leads to models with fewer constraints and variables compared to approaches previously exposed in literature. A similar time-space network structure was introduced for the multiple depot vehicle scheduling problem in public transport by Kliewer et al. [12].

Let \( \mathcal{T} = \{1, 2, \ldots, n\} \) be the set of \( n \) timetabled trips and \( \mathcal{D} = \{1, 2, \ldots, m\} \) be the set of depots. We assume that each trip has exactly two relief points: one at the beginning and another at the end. The set of trips that can be serviced from depot \( d \in \mathcal{D} \) is denoted by \( T^d \). For each depot \( d \in \mathcal{D} \) we define an acyclic vehicle scheduling network \( A^d = (N^d, A^d) \) as described in [7] with \( N^d \) as the set of nodes and \( A^d \) as the set of arcs. We denote \( \tilde{A}^d \subset A^d \) as the set of arcs that requires both vehicle and crew activities. Let \( A^d(t) : \mathcal{T} \to A^d \) be a function that returns the set of trip arcs \( (i, j) \in A^d \) for trip \( t \in \mathcal{T} \) and depot \( d \in \mathcal{D} \). Note that \( A^d(t) = \emptyset \) if \( t \) cannot be operated from depot \( d \). With each arc \( (i, j) \in A^d \) we associate vehicle cost \( c_{ij} \) that is typically a function of travel and idle time. Moreover, we put a fixed (asset) cost for using a vehicle from depot \( d \) on each circulation arc. The maximum capacity \( u_{ij}^d \) of pull-in/out and trip arcs \( (i, j) \in A^d, \forall d \in \mathcal{D} \) is set to 1 while all other arcs have a maximum capacity \( u_{ij}^d \) equal to the number of vehicles available in depot \( d \in \mathcal{D} \). Finally, we introduce two types of decision variables: flow variables and duty variables. Flow variable \( y_{ij}^d \) indicates whether arc \( (i, j) \in A^d \) is used and assigned to depot \( d \in \mathcal{D} \) or not. Likewise, the binary duty variable \( x_k^d \in K^d \) with associated cost \( f_k^d \) indicates whether duty \( k \) is selected for depot \( d \in \mathcal{D} \) or not. Furthermore, \( K^d \) denotes the set of duties that can be operated from depot \( d \in \mathcal{D} \) while \( K^d(i,j) \subset K^d \) defines the set of duties covering arc \( (i, j) \in \tilde{A}^d \). The integrated vehicle and crew scheduling problem with multiple depots (MDVCSP) can be stated as follows:

\[
\begin{align*}
\min \quad & \sum_{d \in \mathcal{D}} \sum_{(i,j) \in A^d} y_{ij}^d c_{ij}^d + \sum_{d \in \mathcal{D}} \sum_{k \in K^d} x_k^d f_k^d \quad (1) \\
\text{subject to} \quad & \sum_{d \in \mathcal{D}} \sum_{(i,j) \in A^d} y_{ij}^d = 1 \quad \forall t \in \mathcal{T} \quad (2) \\
& \sum_{d \in \mathcal{D}} y_{ij}^d = \sum_{d \in \mathcal{D}} y_{ij}^d \quad \forall d \in \mathcal{D}, \forall t \in N^d \quad (3) \\
& \sum_{k \in K^d(i,j)} x_k^d = y_{ij}^d \quad \forall d \in \mathcal{D}, \forall (i,j) \in \tilde{A}^d \quad (4) \\
& 0 \leq y_{ij}^d \leq u_{ij}^d, y_{ij}^d \in \mathcal{N} \quad \forall d \in \mathcal{D}, \forall (i,j) \in A^d \quad (5) \\
& x_k^d \in \{0, 1\} \quad \forall d \in \mathcal{D}, \forall k \in K^d \quad (6)
\end{align*}
\]

The objective (1) minimizes the sum of vehicle and crew costs. Constraints (2)-(3) correspond to a multicommodity flow formulation for the vehicle scheduling problem where the set of trip tasks must be partitioned among the depots (2) and flow conservation is ensured for each depot (3). Constraint set (4) establishes the link between vehicle and crew schedule: each arc covered by a vehicle/vehicles must also be covered by the same number of duties assigned to the depot from which the vehicle(s) originate(s). Constraints (5) guarantee that the maximum capacity of the flow variables is satisfied.

It is well known that even for small instances the number of duty variables can easily run into the billions which makes the enumeration of all feasible duties prohibitive. Consequently, these models cannot be solved using standard mathematical programming methods like branch-and-bound within reasonable time.

III. PROBLEM DECOMPOSITION

Our solution approach decomposes the MDVCSP into different subproblems as shown in Figure 1. First, we assign each trip \( t \in \mathcal{T} \) to a depot \( d \in \mathcal{D} \). Thus, we obtain a trip-depot vector \( \theta \in \{1, \ldots, |\mathcal{D}|\}^{|\mathcal{T}|} \) where each trip is assigned to exactly one depot. In a second phase, we compute the optimal solution of each single depot vehicle scheduling problem and, afterwards, we solve a crew scheduling problem for each depot given the vehicle schedule for that depot. The main advantage of this decomposition is that the vehicle scheduling problem with multiple depots is NP-hard unlike the single depot case that appears in our second phase.

![Fig. 1. Problem decomposition](image-url)
In the second phase, we schedule vehicles independently of crews which corresponds to a traditional (sequential) approach to vehicle and crew scheduling. However, we also consider a partial integration similar to Gintner et al. [8] where we allow to recombine parts of vehicle blocks in order to disclose additional flexibility in crew scheduling while preserving vehicle schedule optimality. The additional flexibility often results in vehicle schedules that allow better crew schedules (with less duties) compared to the sequential approach.

Furthermore, vehicle and crew scheduling can be considered in an integrated way for a given trip-depot vector. That is, we solve $|D|$ integrated vehicle and crew scheduling problems with a single depot. Typically, the additional freedom in scheduling vehicles dependent on crews (and vice versa) leads to better solutions compared to the sequential or partially integrated method.

In summary, there are three different methods to construct a feasible vehicle and crew schedule for a given trip-depot vector: sequential, partially integrated, and fully integrated. Of course, there is a strong relationship between the level of integration and the computational time needed to solve the corresponding problems. The computational time increases with the level of integration. The evolutionary algorithm we propose in the next section is based on this decomposition approach where we first make a trip-depot assignment.

IV. A HYBRID EVOLUTIONARY ALGORITHM

Our evolutionary algorithm (EA) is based on a non-binary representation that is equal to the trip-depot vector $\theta$ from the previous section. A chromosome is a string of length equal to the number of trips where the $i$-th entry contains (the index of) the depot the $i$-th trip is assigned to. We use an evolutionary algorithm to find a good trip-depot assignment where the fitness of a chromosome (individual) is evaluated using mathematical programming techniques. In Section IV-B we describe three different methods to calculate the fitness of an individual. In particular, we use the all-purpose MIP solver ILOG CPLEX [11] and column generation in combination with Lagrangian relaxation.

The size of the EA search space with the non-binary representation is $|T|^{|D|}$. Notice that a feasible vehicle schedule can always be constructed from a given trip-depot vector since each trip can always be covered by a short vehicle block: pull-out trip - service trip - pull-in trip. Furthermore, duty constraints in practice are such that almost all vehicle blocks can be covered by a feasible crew duty. As a consequence, virtually all chromosomes represent feasible trip-depot assignments and, thus, we do not use a repair mechanism to transform infeasible to feasible solutions. However, if a trip-depot assignment $\theta$ does not yield both feasible vehicle and crew schedules, the individual will be discarded after the evaluation of the fitness function. Moreover, the search space can be reduced if the number of trips assigned to a depot must be greater or equal a lower limit $\rho < |T|$. Assigning a small number of trips to a depot often leads to inefficient vehicle and crew schedules since many deadheads and/or vehicles are needed to cover trips that are long way/time away from each other.

In the following, we will describe the essential components of our evolutionary algorithm.

A. Initialization

In the first step of the algorithm an initial population is generated to serve as seed for the evolutionary process. We create our solutions (1) in areas where good solutions are likely to be found and (2) randomly in order to cover a wide range of the solution space. We apply four heuristics of the first category that analyze the geographical structure of the problem, i.e. the start and end location of the service trips. The first three heuristics have been proposed by de Groot and Huisman [6] in combination with a heuristic to split large problem instances such that each split problem can be solved with an integrated approach.

- Assign a service trip to the depot closest to its start location,
- Assign a service trip to the depot closest to the combination of start and end location,
- Solve the multiple depot vehicle scheduling problem and assign a service trip to the depot where it is assigned to in the optimal solution,
- Assign a service trip to the depot either closest to its start or end location.

The rationale behind these heuristics can be understood in the following way. If the trips assigned to the same depot are operated in the same geographical area, it is likely that many of these trips can be covered without extensive deadheading. Furthermore, few deadheads result in a low unproductive overhead since a vehicle and driver outside of the depot spend most of the time on transporting passengers. As stated earlier, we require that at least $\rho = 10$ trips are assigned to a depot. However, if one of the heuristics above leads to an assignment violating the minimum assignment, we randomly shift these trips to other depots.

B. Fitness Calculation

As described in Section III there are three different methods of constructing vehicle and crew schedules for a given trip-depot assignment: sequential, partially integrated, and fully integrated. We apply mathematical programming techniques in order to assess the quality of a particular trip-depot assignment.

1) Sequential and partially integrated evaluation: In the sequential and partially integrated method, we solve a single depot vehicle scheduling problem for each depot with the network simplex implementation of CPLEX [11]. The solution for each depot $d \in D$ consists of a set of arcs $A^d \subset A^\theta$ and the corresponding flow values. Thus, the total fitness $f^d_\theta$ of the vehicle schedule of an individual $\theta$ reads:

$$f^d_\theta := \sum_{d \in D} \sum_{(i,j) \in I^d} y_{ij}^d c_{ij}.$$  

Now, we construct a crew schedule based on the vehicle schedule either sequentially or partially integrated. Although
we omit the mathematical details, in both cases model MDVCSP reduces to a separate, generalized set partitioning problem (SPP) for each depot. If the vehicle schedule is given (the \( y \) variables are fixed) only constraints (4) and (6) remain where the right-hand side of (4) is a constant \( p_{ij}^d \) equal to the flow value of \( y_{ij}^d \). Our computational experiments indicate that it suffices to compute a lower bound on the SPP instead of constructing a feasible crew schedule for each individual.

Since the number of duty variables can be vast even for small-sized problems, we apply a column generation algorithm to compute a lower bound of the SPP. Traditionally, column generation (see Luebbecke and Desrosiers [15] for an extensive survey) is a method to solve linear programs that involve a large number of columns. Instead of solving a large problem with all feasible columns (duties), a sequence of restricted master linear programs (RMP) is solved where each problem contains only a small subset of all columns. However, in our computational experiments we found it very promising to solve the master problem with Lagrangian relaxation (with constraints (4) relaxed) instead of the linear relaxation. That is, we solve the Lagrangian dual with a subgradient method in order to obtain an approximate dual solution and a lower bound for the current RMP. The dual information \( \pi_{ij}^d \geq 0, d \in D, (i, j) \in \mathcal{A} \) is used to price out new columns that are added to the RMP. To solve the pricing problem we set up a duty generation network and solve an associated resource constrained shortest path problem with a dynamic programming algorithm. The column generation process iterates until no new columns can be found. Finally, we end up with \( |D| \) separate sets of columns and an approximate solution to the Lagrangian dual (i.e. the lower bound) for each depot \( d \in D \):

\[
z^d := \max \left\{ \min \sum_{k \in K^d} x_{ik}^d \left( f_k^d - \sum_{(i,j) \in \mathcal{A}^d(k)} \pi_{ij}^d \right) \right. \\
+ \sum_{(i,j) \in \mathcal{A}^d} \pi_{ij} p_{ij}^d x_{ij}^d \in \{0,1\} \}
\]

Furthermore, the overall fitness \( f_\theta \) for an assignment \( \theta \) is defined as the sum of vehicle fitness and crew fitness for each depot:

\[
f_\theta := f_\theta^v + \sum_{d \in D} z^d.
\]

Notice that the set of columns that we obtain while solving the Lagrangian dual for each depot can be used to construct a feasible integer solution (feasible crew schedule). We use the branch-and-bound implementation of CPLEX to generate integer solutions. However, it can be quite time consuming to construct an integer solution for each individual. Therefore, we only compute an integer solution in the final phase of our EA for individuals with a good overall fitness. The integrality gap between the lower bound and the final integer solution was always less than 2%.

Finally, we would like to mention that both sequential and partially integrated fitness calculation do not require an elaborate solution method for integrated problems (such as Borndoefer et al. [5] or Huisman et al. [9]). Instead, we only need a sequential vehicle and crew scheduling algorithm that is used in most commercial software packages for public transport companies. The partially integrated evaluation only differs in the way duties are generated in the column generation pricing problem (but is essentially a set partitioning model as in the sequential crew scheduling problem). Of course, a sequential approach is much easier to implement than a fully integrated one.

2) **Fully integrated evaluation**: So far, we have defined the fitness for a sequential and partially integrated evaluation of a trip-depot assignment. Now, we will specify a fully integrated evaluation. For a given trip-depot assignment model MDVCSP reduces to a minimum cost flow problem in combination with a set partitioning problem. Although this is an NP-hard problem, the vehicle scheduling subproblem can be solved in polynomial time. Again, we first compute a lower bound in order to assess the quality of trip-depot assignment.

We relax constraints (2) and (4) in a Lagrangian way and use a column generation algorithm similar to the one described earlier. The Lagrangian subproblem constitutes a single depot vehicle scheduling problem (with flow conservation constraints (3)) that has to be solved in each iteration of the subgradient method. Notice that the Lagrangian subproblem in the previous subsection was a trivial selection problem since no constraints remained after relaxing (4). Again, we end up with \( |D| \) separate sets of columns and an approximate solution to the Lagrangian dual (i.e. the lower bound) for each depot \( d \in D \). As described earlier the sets of columns can be used to compute an integer feasible solution.

However, it turned out that the computational time to generate an integer solution with a fully integrated model can be prohibitive. On the other hand, the lower bound derived from a fully integrated evaluation gives a better indication on the quality of an assignment than the methods described earlier. This is due to the fact that in this approach vehicles are scheduled dependent on crews (and vice versa).

Therefore, we use the fully integrated fitness calculation only in the first phase of our algorithm. That is, after creating the initial population we use the fully integrated method for fitness assignment in the first \( \lceil |T|/3 \rceil \) iterations of the EA. Later we recalculate the fitness of each individual and iterate using one of the other methods. Our tests indicate that this improves the overall quality of the population.

C. Genetic Operators

1) **Parent Selection**: The parents are selected based on the tournament selection discussed in Beasley and Chu [2]. In tournament selection two pools of randomly selected individuals from the population are constructed. In a second step, we choose the individual with the best fitness from each pool. Our computational tests indicate that forming two pools with 2 individuals each performs best (binary tournament selection).

2) **Recombination**: The recombination performed is based on the fusion operator proposed by Beasley and Chu [2]. The
fusión operator produces a single child and takes both the structure of the parents and their fitness into account. The basic idea is to copy an assignment (gene) to the child if both parents assign the trip to the same depot. If the gene values are different in both parents, it is more likely to inherit the gene from the parent with the better fitness. We compare two parents $\theta_1$ and $\theta_2$ and apply the following rules to obtain child $\theta'$:

$\theta_1[i] = \theta_2[i]$ then $\theta'[i] := \theta_1[i] = \theta_2[i]$

$\theta_1[i] \neq \theta_2[i]$ then $\theta'[i] := \begin{cases} 
\theta_1[i] \text{ with prob. } p := \frac{f_{\theta_1}}{f_{\theta_1} + f_{\theta_2}} \\
\theta_2[i] \text{ with prob. } 1 - p.
\end{cases}$

3) Mutation: Our computational test showed that the quality of the final solution is very sensitive to the mutation rate. In particular, a high mutation rate almost always led to a bad solution quality. Therefore, we use mutation only to eliminate duplicate individuals.

The evaluation of the fitness of an individual is the most time consuming step in our algorithm. Thus, we store all individuals that have been evaluated before. If we generate a child in the recombination phase that has been constructed before, we randomly shift trips between non-empty depots until a new individual is generated. Tests showed that in most cases 2 trip shift suffice to create a new individual.

D. Termination

We have two different termination criteria. The most obvious is to terminate when a given time limit is exceeded. Furthermore, we terminate whenever there has been no significant improvement of the fitness of the currently best individual within the last 3$|T|$ iterations.

V. COMPUTATIONAL RESULTS

We tested our approach on randomly generated benchmark instances that are available at [10]. Our test scenario involved four problem classes. Each class contains 10 instances with 80, 100, 160, and 200 trips involving 4 depots. We make the same assumptions and use the same duty constraints as stated in Borndoerfer et al. [5] and Huisman et al. [9]. All tests were performed on an Intel Pentium 4 2.25GHz/2GB personal computer.

In all tests we set the population size to 20 and the number of iterations is 50. We also tested other settings, but the configuration above appears to be very insensitive to the problem size. The CPU time limit is set to 30 minutes for instances with 80 to 100 trips, to 60 minutes for 160 trips, and 120 minutes for 200 trips. In the following, all results given for our evolutionary algorithms correspond to the average of five runs.

First, we compare our EA methods with the traditional sequential approach to vehicle and crew scheduling. However, we do not use the fully integrated evaluation in the first $\lceil|T|/3\rceil$ iterations of the EA. Consequently, we do not require an integrated solution method (such as Huisman et al. [9]). We rather apply a method similar to the traditional sequential approach to compute the fitness of the individuals.

Table I reports the average solution values for each problem class for the sequential approach (vehicle first - crew second - SEQ), the EA with the sequential fitness calculation (EA-S), and the EA with partially integrated fitness calculation (EA-PI). For each solution approach the number of vehicles, drivers, and the sum of vehicles and drivers (v+d) is given. Furthermore, we present the total CPU time in seconds for computing the solution of the sequential approach and the number of evaluations (#eval) for both evolutionary algorithms.

In Table II we give the relative deviation in percent of the EA solutions from the sequential approach.

It is easy to see that both evolutionary approaches significantly reduce the number of duties between 5.1% and 10.0%. Although the number of vehicles is slightly higher in EA-S and EA-PI, the total number of vehicles and drivers can be decreased between 2.8% and 7.1%. Furthermore, crew costs generally dominate vehicle costs in practice (see Bodin et al. [4]) and, thus, the total savings will be even higher than the numbers indicate. Finally, EA-S appears to perform better than EA-PI. One reason may be that the number of evaluated individuals is much higher in EA-S. To sum up, the EA approaches use an essentially sequential approach for fitness evaluation, but outperform a stand-alone sequential approach.

Next, we compare EA-S with fully integrated approaches from literature. In Table III we report the average results of
the evolutionary algorithm EA-S* where we use the fully integrated evaluation in the first \( \lceil T/3 \rceil \) iterations, Huisman et al. [9] (HFW), and Borndoerfer et al. [5] (BLW). We do not consider the results of Mesquita and Paia [17] here since they use different assumptions. In particular, Mesquita and Paia use a different set of duty types and allow drivers to change their vehicle within a piece of work. We give the total number of vehicles and drivers \((v+d)\), the average computation time \((CPU)\), and, for the EA, the average standard deviation \((\text{avgsdev})\) of the sum of vehicles and drivers. HFW executed their test on an Intel Pentium III 450MHz/128 MB while all tests of BLW were run on an Intel Dual Xeon 3GHz/4GB personal computer. Notice that CPU times cannot be directly compared.

### TABLE III

<table>
<thead>
<tr>
<th>Trips</th>
<th>80</th>
<th>100</th>
<th>160</th>
<th>200</th>
</tr>
</thead>
<tbody>
<tr>
<td>EA-S*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(v+d)</td>
<td>32.1</td>
<td>37.4</td>
<td>49.2</td>
<td>60.8</td>
</tr>
<tr>
<td>avgsdev</td>
<td>0.38</td>
<td>0.36</td>
<td>0.35</td>
<td>0.36</td>
</tr>
<tr>
<td>CPU</td>
<td>00:30</td>
<td>00:30</td>
<td>01:00</td>
<td>02:00</td>
</tr>
<tr>
<td>HFW</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(v+d)</td>
<td>29.6</td>
<td>36.2</td>
<td>49.5</td>
<td>60.4</td>
</tr>
<tr>
<td>CPU</td>
<td>00:13</td>
<td>00:23</td>
<td>02:23</td>
<td>03:00</td>
</tr>
<tr>
<td>BLW</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(v+d)</td>
<td>29.6</td>
<td>35.7</td>
<td>47.7</td>
<td>59.0</td>
</tr>
<tr>
<td>CPU</td>
<td>00:13</td>
<td>00:21</td>
<td>00:44</td>
<td>01:46</td>
</tr>
</tbody>
</table>

It can be seen that our algorithm performs worse than the best known fully integrated algorithm BLW, but the solution quality of EA-S* increases with the problem size. Furthermore, we can conclude that EA-S* is competitive with the integrated approach of HFW for instances with 160 and 200 trips, respectively. In particular, EA-S* was able to find better solutions than HFW for the 160 trips class.

### VI. Conclusions

We have suggested a hybrid evolutionary algorithm to tackle multiple-depot integrated vehicle and crew scheduling problems in public transport. The evolutionary algorithm uses Lagrangian heuristics based on column generation to compute the fitness of the individuals. The algorithm is novel since an evolutionary algorithm has not been applied to integrated vehicle and crew scheduling problems before.

The algorithm is based on a problem decomposition that first assigns trips to depots and, thus, reduces the multiple-depot integrated problem to several integrated problems with a single depot. Unlike the multiple-depot case the single depot case has a vehicle scheduling subproblem that can be solved in polynomial time.

The results reported in the previous section indicate that medium-sized problem instances with multiple depots can be solved by using the proposed evolutionary algorithm. Furthermore, our approach discloses significant savings compared to the traditional sequential approach without requiring a fully integrated solution method. Although our algorithm performs worse than the best known integrated algorithm, it proved to be competitive with other integrated approaches from literature especially for larger instances.

In addition to partitioning trips among the depots, trips must be assigned to vehicle blocks and crew duties. A current limitation of our approach is that we do not take this assignment into account. Further research will focus on how to partition the trips assigned to a depot among vehicles and drivers with a local search heuristic.

### REFERENCES


