Friberg, Christian; Haase, Knut

An exact algorithm for the vehicle and crew scheduling problem

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 416

Manuskripte
aus den
Instituten für Betriebswirtschaftslehre
der Universität Kiel

Nr. 416

An Exact Algorithm for the Vehicle and Crew Scheduling Problem

Christian Friberg and Knut Haase

Manuskripte
aus den
Instituten für Betriebswirtschaftslehre
der Universität Kiel

Nr. 416

An Exact Algorithm for the Vehicle and Crew Scheduling Problem

Christian Friberg and Knut Haase

28. Oktober 1996

Christian Friberg
Institut für Informatik und Praktische Mathematik, Christian–Albrechts–Universität zu Kiel, Preußerstraße 1-9, 24105 Kiel, Germany
Knut Haase
Lehrstuhl für Produktion und Logistik, Institut für Betriebswirtschaftslehre, Christian–Albrechts–Universität zu Kiel, Olshausenstraße 40, 24118 Kiel, Germany
email: Friberg@ramses.informatik.uni-kiel.de
email: Haase@bwl.uni-kiel.de
URL: http://www.wiso.uni-kiel.de/bwlinstitute/Prod
ftp://ftp.wiso.uni-kiel.de/pub/operations-research

Zusammenfassung

We present a model for the vehicle and crew scheduling problem in urban public transport systems by combining models for vehicle and crew scheduling that cover a great variety of real world aspects, especially constraints for crews resulting from wage agreements and internal regulations. The main part of the model consists of a set partitioning formulation to cover the desired trips of the schedule. Because of the great number of columns, e.g. more than 5 million for a problem with 30 trips, a column generation algorithm is implemented to use all columns implicitly for the calculation of the continuous relaxation of the set partitioning problem. The column generation algorithm is embedded in a branch and bound approach to generate an exact solution for the problem. To generate even better lower bound, polyhedral cuts basing on clique detection and a variant of the column generation algorithm that suits the cuts were tested.

(Crew Scheduling, Vehicle Scheduling, Column Generation, Set Partitioning, Polyhedral Cuts)

1 Introduction

A main problem in urban public transport is the scheduling of vehicles and crews to serve trips at minimal cost. Even though public transport companies have for many years solved problems of this type manually, the use of operations research methods is increasing.

The usual procedure in the planning process is to solve the vehicle and the crew scheduling problem sequentially, i.e. first assign the busses to the trips and then assign the drivers to the busses. This approach is strongly criticized by Bodin et al. [Bodin et.al. 1983], because in most cases the crew costs dominate the vehicle costs, and so it should be useful to consider the crew scheduling when the vehicle scheduling is being done. An example of such a relationship can be a vehicle that goes back to the depot instead of reaching the next trip because it is cheaper for the driver who has finished his workday to transfer the bus to the depot and leave it to his follower, than changing at the end of the trip and going to the depot on foot.

A first idea to modify the solution process is to schedule crews first. Unfortunately, this is not a good idea either, because it can increase the cost for vehicles dramatically. The greatest lack of this approach is the increase in the **number** of vehicles. In a first-crew-then-vehicle approach more busses can be needed to serve the crew schedule. Because of the high fixed cost of a vehicle this can be very expensive.

To solve these problems several approaches to schedule vehicles and crews simultaneously have been proposed (see [Ball et.al. 1983] and [Patrikalakis and Xerocostas 1990]). The simultaneous formulation is known as the vehicle and crew scheduling problem (VCSP). Because of the great complexity of the problem the cited publications are all heuristics.

In [Freling et al. 1995] Freling, Boender and Paixão gave the only mathematical formulation for exact solutions of the VCSP we are aware of. They propose a model similar to the VCSP of this work, but using a quasi-assignment problem instead of set partitioning to describe the vehicle scheduling. To calculate the lower bound, two algorithms are presented. One approach uses a two step column generation method to solve the crew scheduling part while the other one is a Lagrangian relaxation without column generation. Both are tested with an example composed of 7 trips and without an embedding in a branch and bound scheme.

All those articles mentioned do not consider the complexity of real crew scheduling problems such as wage agreements and internal regulations, or extensions like multiple depots. For those problems many solutions for either crew **or** vehicle scheduling are published (see e.g. [Desrochers and Soumis 1989] and [Ribeiro and Soumis 1994]). In this paper we want to give a formulation for the vehicle and crew scheduling problem that combines those approaches and so covers most aspects of reality and we will give algorithms to find an optimal solution. To handle the great amount of variables and to model the different types of constraints the vehicle and crew scheduling problem is formulated as a set partitioning problem where the columns represent different workdays of crews or vehicles. The solution is then obtained in a column generation approach to solve the linear relaxation in the nodes of a branch and bound scheme. The subproblems to generate columns appear as resource restricted shortest path problems on scheduling graphs.

To obtain even better lower bounds for the branch and bound algorithm, an approach basing on polyhedral cuts for the set partitioning problem as described by Hoffman and Padberg [Hoffman and Padberg 1993] is taken. As these cuts do not easily match with the column generation algorithm, an additional branch and bound algorithm to generate the new pivot column is presented.

The rest of the paper is organized as follows: In Section 2 a description of the vehicle and crew scheduling problem in a set partitioning formulation is given. Section 3 presents an overview of the solution algorithms, especially the branch and bound scheme. In Section 4 the solution of the continuous relaxation of the VCSP by a column generation algorithm with suitable subproblems is shown. To get even better lower bounds, we use an algorithm to generate polyhedral cuts as defined in Section 5. Section 6 presents the computational results.

2 The Vehicle and Crew Scheduling Problem (VCSP)

To cover a great variety of real world aspects, we use the approach of [Desrochers and Soumis 1989] for crew scheduling. The most important part of [Desrochers and Soumis 1989] is their careful modeling of the crew scheduling graph for the subproblem of the column generation algorithm. An important decision is whether the driven parts are assigned to the nodes (as in [Ball et.al. 1983]) or to the arcs (as in [Desrochers and Soumis 1989]). In this work we will follow the approach of Desrochers et al. [Desrochers and Soumis 1989] because it facilitates the solving of the crew scheduling problem which is the harder part of the two. As the vehicle scheduling is the easier part, we will describe it before the crew scheduling.

2.1 Timetable and Vehicle Scheduling Graph

The obvious information that can be seen after the strategic planning is that part of the timetable that is handed out to the public. Basically it is a collection of *trips* which are defined as the serving of a *line* at a certain time. Therefore the parameters of a trip are the *beginning* of the trip (BT), the *end* of the trip (ET), and the halting points, all determined by their time. Some of the stops are called *relief points* (RP) because the crews are allowed to change. Note that we can take the beginning and the end of each trip as a relief point, too.

The interesting pieces of a trip are those that have to be served by the vehicle and the crew without changes. These parts from one relief point to the following are called *dtrips* (as in [Ball et.al. 1983]). In crew scheduling those parts are often called *tasks*, see e.g. [Desrochers and Soumis 1989]. We define for every trip i a corresponding sequence of relief points $\mathrm{rp}_{ij}, 0 \leq j \leq n_i$, where rp_{i0} and rp_{in_i} denote the beginning and the end of the trip, respectively. The relief points rp_{ij-1} and rp_{ij} define dtrip j for all j, each supplied with the costs for a vehicle vc_{ij} and a crew cc_{ij}. Note that the costs are given and not calculated from the trip itself in any way. This will become important in Section 4, where we can assume them to be marginal costs instead of real ones. The parts driven between relief points and transfers to or from the depot will be called **travels** in this paper. The starting and the ending time of the dtrips can be taken from the complete timetable.

To complete the required arguments for the timetable, we need to know several supplementary parameters: For all vehicles and crews the costs and the times to go from the depot to the beginning of every trip i ($vcdb_i$, $ccdb_i$, tdb_i), from the end of every trip i to the depot ($vced_i$,

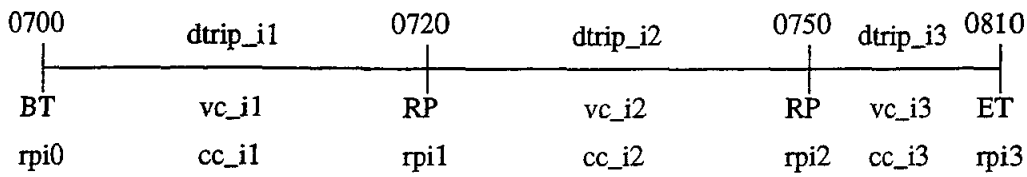

Abbildung 1: Trip in a timetable

$cced_i$, ted_i), from the end of trip i to the beginning of trips j ($vceb_{ij}$, $cceb_{ij}$, teb_{ij}), plus the waiting costs at the depot between two trips i and j for vehicles ($vwcd_{ij}$) and for crews ($cwcd_{ij}$) or two relief-points rp_{ij} and rp_{kl} for crews $cwcd_{ijkl}$).

Additionally for the crew scheduling (defining all begin and end points as relief points) the costs and times to go from the depot to every relief point rp_{ij} ($cdrp_{ij}$, $tdrp_{ij}$), from every relief point rp_{ij} to the depot ($crpd_{ij}$, $trpd_{ij}$), from relief point rp_{ij} to rp_{kl} for every possible combination (crp_{ijkl}, trp_{ijkl}). These costs can be interpreted as costs needed to get a crew from one point to another without driving a vehicle. Finally we need the constant costs for vehicles and crews to start and finish the day (e.g. fixed costs of a vehicle, basic payment for crews) which will be termed as **vsignon**, **vsignoff**, **csignon**, and **csignoff**.

Now we can derive the graphs for vehicles and crews from the timetable taking into consideration additional constraints. If we assume a homogeneous fleet of vehicles at a single depot with no extra constraints on their use (e.g. refueling) we can easily define a network for the vehicle scheduling, just reducing the information of the timetable. At first we have to add the costs of all dtrips to get the aggregated cost of the trips

$$vc_i := \sum_{j=1}^{n_i} vc_{ij}$$

and then define a graph where the nodes are beginning and end of trips, arrival or departures at the depot and the start and the end of the day, and where the arcs are annotated with the costs from the timetable. With the selection of the arcs we can model obvious constraints, like no vehicle can go back in time to serve two trips simultaneously.

Definition 2.1 (vehicle scheduling graph) Let $VG = (VN, VA)$ be a graph with nodes

$$VN = \{\text{start, end}\} \cup \{\mathrm{bt}_i, \mathrm{et}_i, \mathrm{bdepot}_i, \mathrm{edepot}_i \mid \forall\ \mathrm{trip}_i\}$$

where start and end are the source and the sink of the network, respectively. The arcs in VA are defined by the feasible stages supplied with their costs. The feasibility is defined relating to the time constraints, i.e. a vehicle can not leap back into time and should not exceed a maximal waiting time *vwmax*.

The arcs of the vehicle scheduling graph are (start,bdepot$_i$) with cost vsignon, (bdepot$_i$,bt$_i$) with cost $vcdb_i$, (bt$_i$,et$_i$) with cost vc_i, (et$_i$,edepot$_i$) with cost $vced_i$, (edepot$_i$,end) with cost vsignoff, (et$_i$,bt$_j$) with cost $vceb_{ij}$ if $teb_{ij} \in [0, vwmax]$ and (edepot$_i$,bdepot$_j$) with cost $vwcd_{ij}$ if $(tdb_j + ted_i) \in [0, vwmax]$.

The last two sorts of arcs belong to direct or indirect (over the depot) stages, respectively, from one trip to another. Note that the times for these stages may be negative if a trip starts before the other one begins. So the constraints in building the arcs ensure that all times are positive and do not exceed a maximal waiting time. For every arc a its cost will be denoted $c(a)$. □

The depot nodes corresponding to the beginning and the end of each trip are used to model a vehicle starting from or coming back to its depot at a certain point of time. The importance of this approach will be seen in the combination with the crew scheduling.

To illustrate the definition we give a small example: In the graph in Figure 2, a vehicle is

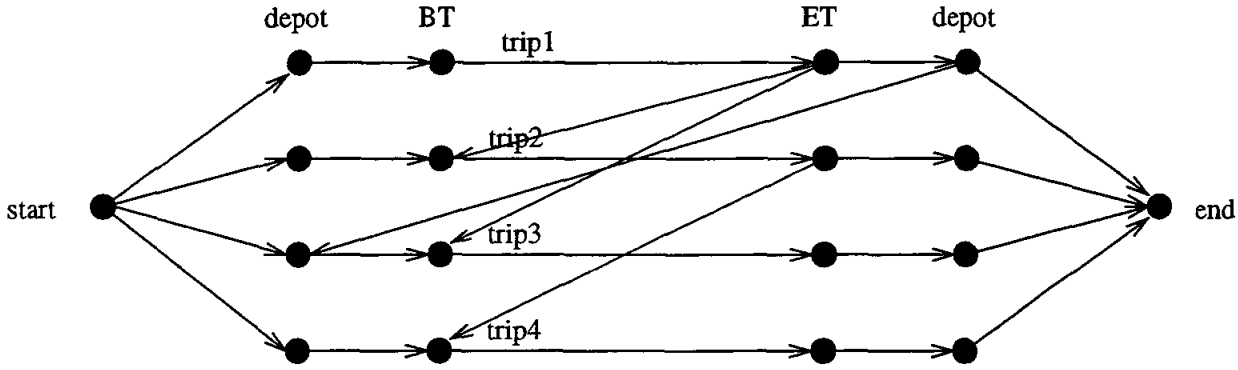

Abbildung 2: Example of a vehicle scheduling graph

allowed to start the day with trip 1, come back to the depot afterwards to go to trip 3 and then close the day. Another possibility is to go from the end of trip 1 **directly** to the beginning of trip 2 or 3. After serving this trip one can decide to go directly to trip 4 or close the day by driving back to the depot. Note that there is no arc from the end of trip 1 to the beginning of trip 4 even though both can be served by one vehicle if trip 2 is taken between them. This can be explained by an expansion of the waiting time between those trips.

In the remainder of the work it will be important to know the schedule for **one** vehicle during a day. This is exactly what is described with a path from node *start* to node *end*. Such a path is called a **block** (see [Desrochers and Soumis 1989]).

Definition 2.2 (block) A *block* b in the vehicle scheduling graph VG is defined as a finite sequence $node_0, \ldots, node_{r_b}$ of nodes $\in VN$, where $node_0$ = start, $node_{r_b}$ = end and

$(node_i, node_{i+1}) \in VA$ for all i. Its *cost* is defined as

$$c(b) = \sum_{i=0}^{r_b-1} c((node_i, node_{i+1})).$$

A *cost minimal block* is a block b such that $c(b) \leq c(b')$ holds for every other block b'. □

2.2 The Crew Scheduling Graph

In [Desrochers and Soumis 1989] and [Desrochers et.al. 1990] the authors propose a clever crew scheduling graph that includes resources and special subsumptions of arcs to cover a great variety of real world constraints. To combine their approach with the vehicle scheduling of the previous subsection, we will formalize it in two steps: first a simple crew scheduling graph as an extension of the vehicle scheduling graph and then an improvement of it, covering all aspects of [Desrochers and Soumis 1989], with an exact formal relationship between them.

In a first approach the crew scheduling graph can be described in a similar way as the vehicle scheduling graph. The only extension necessary is the consideration of the relief points where the crew is allowed to change. At this points the crew can stay at the vehicle, go to another vehicle, go to the depot, or end the workday. A graph containing such information will be called a *simple crew scheduling graph.*

Definition 2.3 (simple crew scheduling graph) Let $SCG = (CN, SCA)$ be a graph with nodes

$$CN = \{\text{start, end}\} \cup \{\text{rp}_{ij}, \text{bt}_i, \text{et}_i, \text{bdepot}_i, \text{edepot}_i, \text{bdepot}_{ij}, \text{edepot}_{ij} \mid \forall\ \text{trip}_i,\ j = 1, \ldots, n_i\}$$

where start and end are the source and the sink of the network, respectively. The different types of depot nodes belong to begin and end of trips (single index) or of travels from relief points (double index). The arcs in SCA are all feasible stages associated with their costs. The feasibility is defined relating to the time constraints, i.e. a crew can not leap back into time and should not exceed a maximal waiting time *cwmax*.

The crew scheduling graph has the following arcs: (start,bdepot_{ij}) with cost csignon, (start,bdepot_i) with cost csignon, (bdepot_{ij},rp_{ij}) with cost $cdrp_{ij}$, (bdepot_i,bt_i) with cost $ccdb_i$ ($\star$), (bt_i,rp_{i0}) with cost 0, (rp_{ij-1},rp_{ij}) with cost cc_{ij}, (rp_{ij},edepot_{ij}) with cost $crpd_{ij}$, (et_i,edepot_i) with cost $cced_i$ ($\star$), (rp_{in_i},et_i) with cost 0, (edepot_{ij},end) with cost csignoff, (edepot_i,end) with cost csignoff, (rp_{ij},rp_{kl}) with cost crp_{ijkl} if $trp_{ijkl} \in [0, cwmax]$, ($\text{edepot}_{ij}$,$\text{bdepot}_{kl}$) with cost $cwcd_{ijkl}$ if$(tdrp_{ij} + trpd_{kl}) \in [0, cwmax]$, ($\text{et}_i$,$\text{bt}_j$) with cost $cceb_{ij}$ if $teb_{ij} \in [0, cwmax]$ ($\star$) and (edepot_i,bdepot_j) with cost $cwcd_{ij}$ if $(tdb_j + ted_i) \in [0, cwmax]$.

The last 4 sorts of arcs belong to direct or indirect (over the depot) stages, respectively, from one relief point to another in a different trip. The first two represent travels of a crew without the vehicle, the third means the transport of the bus while the fourth describes waiting time at a depot. The arcs that are marked with an asterisk ($\star$) combine vehicle and crew scheduling and therefore will be called *linking arcs*. In Section 2.3 it will be enforced that these arcs are taken if and just if the same stage is served by a vehicle, i.e. a crew has to do the **transfer** of this vehicle. For every arc a its cost will be called $c(a)$. □

Even in a simple example we can see how many arcs have to be used to model the description. In Figure 3 two trips with one or two additional relief points, respectively, are connected with

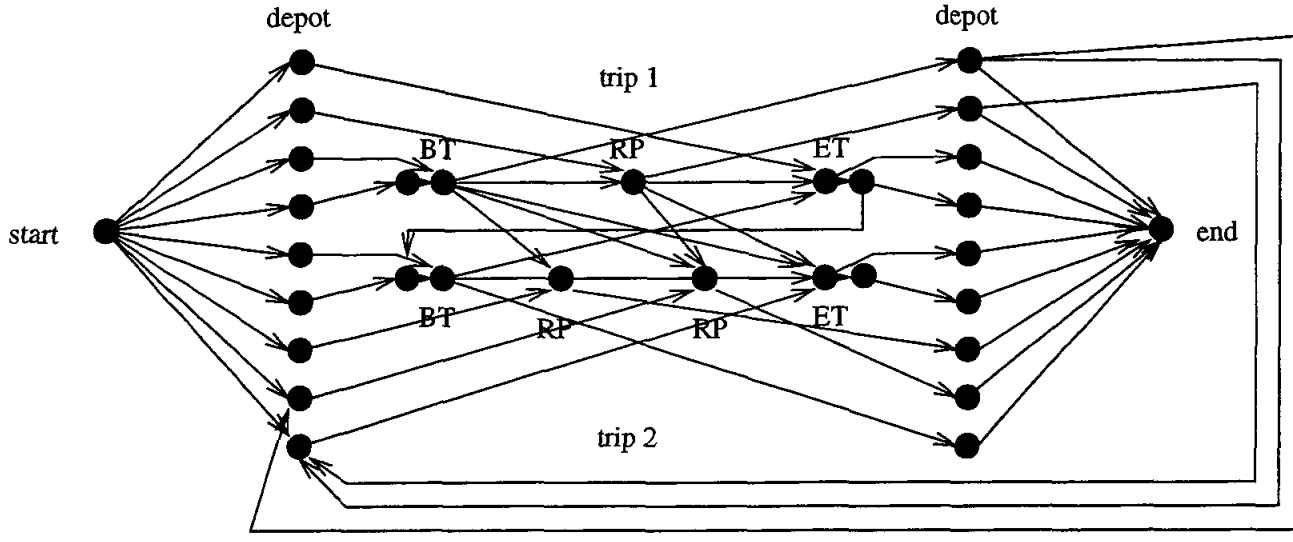

Abbildung 3: A simple crew scheduling graph

the arcs allowed. Every relief point is associated with two depot nodes that model a travel to the depot without a vehicle. The beginning and the end of each trip is connected with a second depot node to describe the stage to the depot with the vehicle. Note that a crew is allowed to drive a vehicle from the end of trip 1 to the beginning of trip 2 but may not go there **alone**, even if an arbitrary relief point is chosen. This can be used to model time differences between driving the vehicle or walking from one relief point to another.

Even if this approach covers most information needed for a simple scheduling there is still a lack of reality in it. Unlike the vehicle scheduling, the crew scheduling is strongly determined by wage agreements and internal regulations. So there are restrictions on the length of a workday, the length of work without a break, or other resources.

There are two ways to model these restrictions in a graph: with path feasibility constraints,

and in the definition of the arcs. Both ways will be used in this approach, following [Desrochers and Soumis 1989].

The first idea used here is the introduction of a *piece of work*. A piece of work is a sequence of tasks done by a crew on the **same** vehicle. So every block can be splitted into several pieces of work. The wage agreements usually restrict the number of pieces on a workday and the number of tasks in a piece, so it is useful to form the graph mainly with such arcs. In Figure 4 we

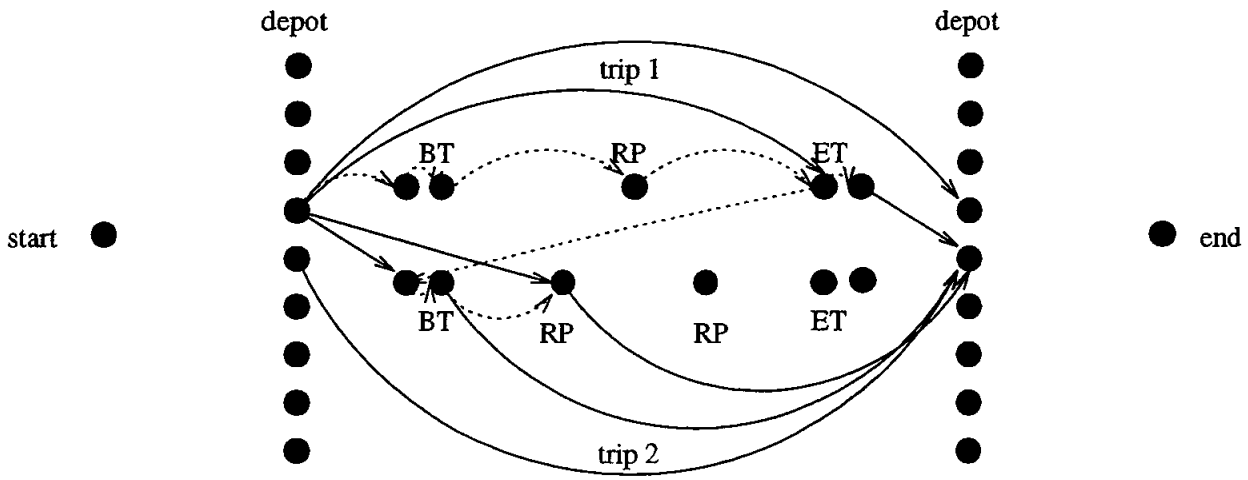

Abbildung 4: Pieces of work

see an example of pieces of work derived from Figure 3, where one piece may consist of 3 to 5 tasks. The graph is still simplified by taking into account that a vehicle must be driven by a crew all the time. So all pieces that arrive at nodes where no arc departs (and vice versa) can be eliminated. Note that for example the arc from the beginning depot of trip 1 to the first relief point of trip 2 symbolizes a piece of work that covers the complete first trip, the stage to, and the first dtrip of trip 2, all served by one vehicle. The corresponding arcs of the simple scheduling graph are represented in a dotted version. In practical situations also the amount of time available for some piece of work will be restricted, so even more arcs can be eliminated.

The remaining arcs are used to model sign on, sign off (all connected with start respectively end), and breaks. In use, most of these arcs can be eliminated because of restrictions such as the length of breaks.

As this approach can model time restrictions of parts of the graph as pieces of work, we still have to integrate global constraints such as the total working time or the number of pieces on a day. These constraints can be modeled as path feasibility constraints. These are formulated as resource constraints at each node and resource consumptions on each arc. Thereby resource constraints are formulated as feasibility windows that restrict the use of the

resource consumptions on the arcs.

To explain this approach we will look at some examples from [Desrochers et.al. 1990]: If the number of pieces of work are restricted to two, each arc representing such a piece has a piece consumption of 1, while all other arcs have a consumption of 0. The start node has the feasibility window $[0, 0]$, while all other nodes have the window $[0, 2]$ for pieces. If the wage agreements guarantee a minimal break time *mbt*, every arc considered to be a break (e.g. the waiting arcs at the depot) has a break time consumption of the time of the arc. The sink of the graph, representing the end of the day, then has a feasibility window $[mbt, \infty]$. Similar constraints e.g. for number of breaks, total working time, etc. can be formulated in a similar way, each associated with another set of resource constraints and consumptions in the graph. Many examples for this approach can be found in [Desrochers et.al. 1990].

With these requirements we can now formulate the crew scheduling graph that will be used in the remainder of the work.

Definition 2.4 (crew scheduling graph) Let $CG = (CN, CA)$ be a graph. CG is called a *crew scheduling graph* if there exists a simple crew scheduling graph $SCG = (CN, SCA)$ with a mapping Φ from CA to the set of **sequences** of SCA, such that for every arc a in CA the sequence $\Phi(a)$ is a **path** in SCG.
The costs of each arc a of CA are determined by

$$c(a) = \sum_{(node_i, node_{i+1}) \in \Phi(a)} c((node_i, node_{i+1}))$$

as the sum of the covered arcs of the simple graph.

CG is called a *resource restricted crew scheduling graph* if there exists a number $r_{CG} \geq 1$ of resources so that for every node we have the resource constraint windows $[a^l_{node}, b^l_{node}]$ and for every arc we have k resource consumptions d^l_k with $1 \leq l \leq r_{CG}$. □

Further we will just use resource restricted crew scheduling graphs and so will call them crew scheduling graphs to simplify the notation. Note that from the current formulation many different combinations of arcs and resources are allowed if they are needed to cover special aspects.

In the remainder of the work it will be important to know the schedule for **one** crew during a day. This is exactly what is described with a path from node *start* to node *end*. Such a path is called a **workday** (see [Desrochers and Soumis 1989]).

Definition 2.5 (workday) A *workday* w in the crew scheduling graph CG is defined as a finite sequence $node_0, \ldots, node_{s_w}$ of nodes $\in CN$ where $node_0 =$ start, $node_{s_w} =$ end and $(node_i, node_{i+1}) \in CA$ for all i.

A workday is *feasible* if for all $t \in \{0, \ldots, s_w\}$ and all $l \in \{1, \ldots, r_{CG}\}$

$$\sum_{i=1}^{t} d^l_{(node_{i-1}, node_i)} \in [a^l_{node_i}, b^l_{node_i}]$$

holds.

Its *cost* is defined as

$$c(w) = \sum_{i=0}^{r_w - 1} c((node_i, node_{i+1})).$$

A *cost minimal workday* is a workday w such that $c(w) \leq c(w')$ holds for every other feasible workday w'. □

2.3 The Set Partitioning Problem

In the last subsections we have seen how the scheduling demands for vehicles and crews can be coded using graphs. Now we have to define the formulation of the optimization problem to get a cost minimal vehicle and crew schedule, which will be given in a binary linear program.

In general this can be done by a *set partitioning* approach. In this approach we have binary variables for every trip or dtrip in the graph and try to minimize the costs while the variables cover the arcs exactly once. A problem that arises in this approach is how to model the constraints that are coded in the graph, such as path constraints or the structure of the networks. So a favored procedure is to define the sets of feasible blocks (vehicle paths, VP) and workdays (crew paths, CP) and to cover the network with them. The problem of generating feasible paths can then be solved by special algorithms using the graphs directly.

For this approach we define binary variables x_i for every feasible vehicle path vp_i, and y_j for every feasible crew path cp_j, where $x_k = 1$ ($y_k = 1$) means that path p_k is used in the solution. To describe the paths in the LP-formulation they are coded with binary constants:

- $vtrip_{ik}$: vehicle path vp_i serves trip k
- $cdtrip_{jl}$: crew path cp_j serves dtrip l
- $varc_{ik}$: vehicle path vp_i serves arc k
- $carc_{jk}$: crew path cp_j serves arc k

where constant=1 means that the corresponding arc is covered by the path. The constants $varc_{ik}$ and $carc_{jk}$ describe the linking arcs in the vehicle and the crew scheduling graph which represent the transfer of a vehicle by a crew. Note that the paths are taken from the crew scheduling graph while the covered arcs originate from the simple crew scheduling graph using a mapping Φ as in definition 2.4. (Look again at the dotted path in Figure 4.)

The formulation of the vehicle and crew scheduling problem as a set partitioning model with additional constraints then is given in the next definition:

Definition 2.6 (vehicle and crew scheduling problem) Let VP and CP be the sets of all feasible vehicle and crew paths.

$$Min \sum_{vp_i \in VP} c(vp_i)x_i + \sum_{cp_j \in CP} c(cp_j)y_j \tag{1}$$

$$\sum_{vp_i \in VP} vtrip_{ik}x_i = 1 \quad \forall \text{ trips } k \tag{2}$$

$$\sum_{cp_j \in CP} cdtrip_{jl}y_j = 1 \quad \forall \text{ dtrips } l \tag{3}$$

$$\sum_{vp_i \in VP} varc_{ik}x_i - \sum_{cp_j \in CP} carc_{jk}y_j = 0 \quad \forall \text{ linking arcs } k \tag{4}$$

$$x_i, y_j \in \{0,1\} \quad \forall\, i,j \tag{5}$$

□

As stated above the objective function (1) is the sum of the costs for vehicle and crew schedules. The constraints (2) and (3) ensure that all trips and dtrips are covered, while restriction (4) links vehicle and crew schedules together so that every vehicle is driven by exactly one crew on the linking arcs. To substantiate the definition of the linking condition, we have to show that if condition (4) is fulfilled every stage is either served by exactly one vehicle and one crew or by none of them.

Remark 2.7 (linking) Let (x^*, y^*) be a feasible solution for the set partitioning model. Then for every linking arc k, $\sum_{vp_i \in VP} varc_{ik}x_i^* = 0$ (and therefore $\sum_{cp_j \in CP} carc_{jk}y_j^* = 0$) or $\sum_{vp_i \in VP} varc_{ik}x_i^* = 1$ (and therefore $\sum_{cp_j \in CP} carc_{jk}y_j^* = 1$).

Proof: Let k be a linking arc. Based on the definition of vehicle scheduling graphs and simple crew scheduling graphs it is evident, that if the first node k is the end of a trip et_i it has not more than one predecessor. (Note that this does **not** hold for an arbitrary crew scheduling graph, but the set partitioning formulation uses the simple version of it.) Because of constraint

(2) (or (3)) and the definition of a path (in 2.2 or 2.5) it is is easy to verify that our claim holds in this case.

So let k be $(\text{bdepot}_i, \text{bt}_i)$. Similar to the other cases the node bt_i has just one successor. Again because of constraint (2) and the definition of a path the statement holds. □

With these definitions we have a useful formulation of the vehicle and crew scheduling problem with a set partitioning approach. In the next subsection we will show some extensions to the problem and how they can be modeled. The remainder then deals with methods to find optimal solutions.

2.4 Extensions

So far we have concentrated on an undoubted complex problem, but there is still a lack of reality in the model. Many of the articles cited above deal with extensions of the model. For a survey of different problems in vehicle scheduling see [Bodin et.al. 1983].

An often analyzed improvement of the vehicle scheduling problem is the multiple depot vehicle scheduling problem. In this formulation all vehicles have to go back to the depot they departed from. Additionally, all the costs on the arcs, including fix costs using sign on, depend on the used depot. Besides the modeling of real depots, this approach is also useful to describe different types of vehicles with different costs, i.e. every type of vehicle is associated with a special depot.

To embed the multiple depots in our current model we can define a set of vehicle scheduling graphs, where each graph carries the costs of a single depot. With this approach we can e.g. force the use of a special type of vehicle on a trip by not including the corresponding arc in the other graphs. The combination of the different schedules is very simple in our set partitioning model. As usual, every path is coded in the constants vtrip and varc and so we do not have to change the LP-formulation. A detailed description of such an approach can be seen in [Ribeiro and Soumis 1994].

One problem of this approach is the combination with the crew scheduling. As we have seen, the vehicles always have to be driven, so we must distinguish between the different depots in the crew scheduling, too. This can be done by adding several depot nodes to the relief points of the graph, one for every vehicle scheduling graph belonging to a certain depot that covers the arc between the relief point and the depot (see Figure 5). Note that there is no arc between the depot nodes because both represent the **arrival** at a depot. Of course there can be arcs between arrival and starting nodes of different depots, so that the same crew can drive different

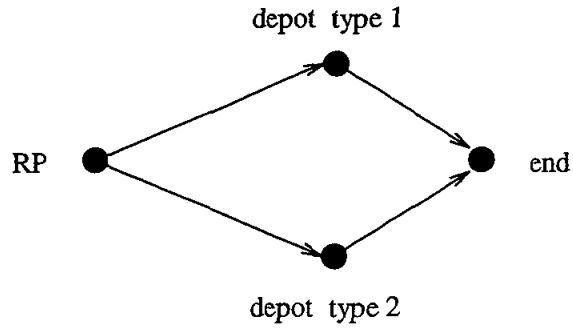

Abbildung 5: Relief point with two depots

types of vehicles.

Other aspects as a restricted number of vehicles or special sorts of workdays, such as trippers, straights and splits and their incorporation in the column generation algorithm are explained in detail in [Ribeiro and Soumis 1994] and [Desrochers and Soumis 1989].

In summary, we combined the most sophisticated approaches for vehicle and crew scheduling into a single problem (VCSP) that covers nearly all real world aspects of the operational planning step of the urban public transport systems. The next sections will demonstrate how such problems can be attacked.

3 The Branch and Bound Algorithm

In this article we present an approach that solves the VCSP within a branch and bound algorithm. The lower bounds are calculated by dropping the integral constraints of the set partitioning formulation and solving the continuous linear program with column generation. The application of this special approach is explained in detail in the following sections, so we will now concentrate on the branching.

Definition 3.1 (branching in the VCSP) Let $S = \{x \mid Ax = b,\ x \in Z^+\}$ be the set of feasible integral solutions of the VCSP as in Definition 2.6. Define for every index set I of columns the set

$$S^I = \{x \in S \mid i \in I \Rightarrow x_i = 0\}$$

Obviously we have $S^I \subseteq S$ for all I and $S^\emptyset = S$. A solution for the continuous relaxation can be found easily by solving the relaxation of S with the simplex algorithm without considering the columns in I.

Now let x be an optimal feasible solution of cx for the relaxation of S^I. Then calculate for

every linking arc $link_i$ its violation

$$v_i \;=\; 0.5 - \mid 0.5 - \sum_{j=1,\ P_j\ block}^{n} x_j a_{i+n_t+n_d j} \mid$$

(with n_t and n_d the number trips and dtrips, respectively) and detect the maximal violation $v_{max} = \max v_i$ with a corresponding linking arc $link_k$. If $v_{max} \neq 0$, then x is not integral, i.e. the problem is not leveled for this branch, and there is at least one column P_j, $j \notin I$ and $x_j \neq 0$ that covers link k. Let I' be $I \cup \{j \neq I \mid P_j \text{ covers link } k\}$ and I'' be $I \cup \{j \neq I \mid P_j \text{ covers link } l, l \neq k, link_l \in Ray(link_k)\}$, where $Ray(a)$ is the set of arcs that have the same starting point as the arc a. Define

$$\rho(S^I) \;=\; \{S^{I'}, S^{I''}\}$$

as the (binary) branching function. □

To justify the definition we have to show that x is integral if $v_{max} = 0$. This can only be shown for the components of x that correspond to a feasible path of the vehicle scheduling graph.

Theorem 3.2 Let S, I, and x be as in Definition 3.1. If $v_{max} = 0$, the solution x is integral in every component x_j, with column P_j representing a **block**.

Proof: Let $v_{max} = 0$, and assume that there exists a column P_j with $x_j \neq 0$ and $x_j \neq 1$ that corresponds to a **block**. Let link l be the last link on the block before the end node. Because $v_{max} = 0$, there exists at least another column P_t with $x_t \neq 0$ and $x_t \neq 1$ that also covers link l. Obviously, the start node of link l represents the arrival at a depot. From the definition of the vehicle scheduling graph it is easy to conclude that the depot has an unique predecessor, the end of a trip, so P_j and P_t both cover this trip. Now we have three different cases:

Case 1 P_j covers a link l' from the end of another trip to the beginning of the current one. As $v_{max} = 0$, we have $\sum_{r=1,\ P_r\ block}^{n} x_r a_{l'+n_t+n_d r} = 1$ and $\sum_{r=1,\ P_r\ block}^{n} x_r a_{l''+n_t+n_d r} = 0$ for all other links l'' that arrive at the beginning node. So P_t covers link l', too.

Case 2 P_j covers a link from a start at a depot to the beginning of the current trip and a link from an arrival at another depot to the current one. Analogous to Case 1 P_t covers both links.

Case 3 P_j covers a link from a start at a depot to the beginning of the current trip and an arc from the start of the day to the depot. As in Case 1 and 2 one can show that P_t covers the arcs.

In Case 3 P_j and P_t are obviously derived from the same block. In Case 1 and 2 a simple induction shows the same conclusion. So we have $P_j = P_t$, which is a contradiction, that shows that the assumption $x_j \neq 0$ and $x_j \neq 1$ is false. □

To extend the approach to get an integral solution for all columns including the workdays, the violation of the **breaks** and **travels on foot** have to be calculated. The branching mechanism and the proof then follow the same lines as for vehicles. Afterwards, a branching function ρ is defined that obviously defines a finite tree whose leaves are the integral solutions of S.

So far we have described how columns that have already been generated can be eliminated. Now we have to consider the explicit columns that are coded into the feasible paths of the scheduling graphs. As the index set I is derived from sets of linking arcs, this can be done by removing arcs from the graph. In the implementation the arcs are simply marked to reuse them in other branches.

4 The Column Generation Approach

As stated in Ribeiro and Soumis [Ribeiro and Soumis 1994], one main advantage of the set partitioning formulation is the observation that the gap between the optimal solution for the integral problem and the relaxed problem is often very small and unnecessary branches are very likely been cut from the bounding rules. So the most important task in developing algorithms for the VCSP is to create efficient solvers for linear programs with respect to the special structure of the VCSP. This subsection presents an overview of the algorithms described in full length in the following sections.

The complexity of the VCSP is determined by the number of arcs, mainly the trips, dtrips, and links, of the scheduling graphs. This number corresponds to the number of rows of the constraint matrix A. Some of them can be dropped, by using a pivot and probe algorithm [Sethi and Thompson 1984] to solve the linear program. Unfortunately, the number of columns is equivalent to the number of **feasible paths** which is increasing exponential on the number of arcs. So the time to search for the pivot column is exponential, too. Moreover, if the number of columns is to big (more than 5 million columns for 30 trips!) they can not be stored in main memory and so reduce the performance of the computation. The column generation algorithm now makes use of the relationship between the columns and the feasible paths of the graphs. Instead of enumerating all columns before the start of the simplex algorithm, only a small initial matrix is taken. Then successively the smaller problems are solved, and instead of searching the pivot column in the large matrix, it is **generated** using a *subproblem* and added to the

small matrix. This iteration is done until the subproblem can not find a negative marginal cost column. At this point the whole linear program is solved under implicit consideration of the complete set of columns.

To generate columns the simplex multipliers of the small linear programs optimal solution are subtracted from the costs of the corresponding trips, dtrips, and links (note that the rows of the matrix are constraints for the partitioning of these arcs). The pivot column of the complete matrix then is equivalent to a **shortest path** in one of the scheduling graphs. If the pivot and probe algorithm is used to solve the linear program, only the multipliers of the active rows are subtracted from the costs of the arcs. The costs that are not represented by an active row are just added to the cost of the path (as are the costs of other arcs like travels on foot or signon/signoff that are neither represented in the set partitioning formulation). In Subsection 4.2 solutions for the shortest path problems of the vehicle and the crew scheduling graphs are represented. If one of the shortest paths is negative, it is translated into a new column of the constraint matrix with entries 1 or -1 as described in Definition 2.6 if the row corresponds to a trip, dtrip, or link.

4.1 Column Generation Principle

The idea of column generation is easy to understand if the revised simplex algorithm is used as its basis (see Lasdon [Lasdon 1970]). In Definition 2.6 the constraint matrix was defined such that every column corresponds to a feasible path of the vehicle scheduling graph, i.e. a block, or of the crew scheduling graph, i.e. a workday. This can be formalized by using two bijections ϕ_v and ϕ_c that map the set of blocks VP to the set of vehicle columns $VC = \{P_1, \ldots, P_{n_b}\}$ and the set of workdays CP to the set of crew columns $CC = \{P_{n_b+1}, \ldots, P_{n_b+n_w}\}$, respectively, where n_b and n_w are the number of blocks and workdays, respectively. Define further the number of trips n_t, dtrips n_d, and linking arcs n_l. The vehicle mapping is defined as

$$\phi_v : VP \mapsto VC,\ \phi_v(b) = P_j :\ a_{ij} = \begin{cases} 1 & if\ 1 \leq i \leq n_t\ and\ \text{trip}_i \in b \\ & or\ n_t + n_d < i \leq n_t + n_d + n_l\ and\ \text{link}_i \in b \\ 0 & otherwise \end{cases}$$

while the crew mapping is

$$\phi_c : CP \mapsto CC,\ \phi_c(b) = P_j :\ a_{ij} = \begin{cases} 1 & if\ n_t < i \leq n_t + n_d\ and\ \text{dtrip}_i \in b \\ -1 & if\ n_t + n_d < i \leq n_t + n_d + n_l\ and\ \text{link}_i \in b \\ 0 & otherwise \end{cases}$$

where just the entry for the linking arcs is different from ϕ_v to couple vehicles and crews.

The main idea of the column generation algorithm is now to replace in the simplex algorithm the calculation of the marginal costs for **all** columns and the search for the minimal marginal cost column (pivot column), by a **generation** of this column, if possible. This is done by first solving a smaller linear program with a submatrix A' of A, using the normal revised simplex method, and then generating a column P_j not in A' with

$$\bar{c}_i \;=\; \min_k \bar{c}_k \;=\; \min_k (c_k - \pi P_k)$$

This generation is done by solving a *subproblem.* As every column corresponds to a feasible path of a scheduling graph, the generation process can be done by a **shortest path problem** where the simplex multipliers π_i are added to the costs of the arcs.

Definition 4.1 (marginal cost graphs) Let $VG = (VN, VA, c_v)$ be a vehicle scheduling graph and $CG = (CN, CA, c_c)$ be a crew scheduling graph with arc costs c_v and c_c as described in Section 2. Further let $\pi_1, \ldots, \pi_m$ be simplex multipliers of the linear program.
Now define the marginal cost graphs $VG' = (VN, VA, c'_v)$ with

$$c'_v(a) \;=\; \begin{cases} c_v(a) - \pi_i & if\ a = \text{trip}_i \\ c_v(a) - \pi_{i+n_t+n_d} & if\ a = \text{link}_i \\ c_v(a) & otherwise \end{cases}$$

and $CG' = (CN, CA, c'_c)$ with the costs

$$c'_c(a) \;=\; c_c(a) - \sum_{dtrip_i \in \Phi(a)} \pi_{i+n_t} + \sum_{link_i \in \Phi(a)} \pi_{i+n_t+n_d}$$

calculated with the **simple** crew scheduling graph (see Definition 2.4 in Subsection 2.2). □

With these definitions we can now formulate the column generation algorithm.

Definition 4.2 (column generation) Let A' be a **nonsingular** submatrix of A and $P(A')$ be the linear program for the VCSP restricted to A' and the corresponding subvector c' of c.

1. Solve the linear program $P(A')$
2. Calculate the marginal cost graphs VG' and CG' as in Definition 4.1. Let b be a shortest feasible path (cost minimal block) in VG' and w be a shortest feasible path (cost minimal workday) in CG'.
3. If both $c'_v(b)$ and $c'_c(w)$ are nonnegative, stop. The current solution is optimal for $P(A)$.

4. If $c'_v(b) < 0$, redefine A' as

$$A' \leftarrow (A', \phi_v(b))$$

and c' as $(c', c_v(b))$. If $c'_c(w) < 0$, redefine A' as

$$A' \leftarrow (A', \phi_c(w))$$

and c' as $(c', c_c(w))$.

5. Continue with Step 1.

In Step 4 the matrix A' can be expanded by more than one column to accelerate the algorithm (multiple pricing). □

Theorem 4.3 The column generation algorithm stops with an optimal solution x' *iff* the simplex algorithm stops with an optimal solution x, and in this case $cx' = cx$ holds.

Proof Let x' be a solution of the column generation algorithm and π be the vector of simplex multipliers after Step 1. Obviously, x' is also a basic feasible solution for $P(A)$. We just have to prove that there exists no column P_j of A with $c_j - \pi P_j$ negative to show that x' is also optimal for $P(A)$. Assume that there exists such a column. Then there exists the feasible path $\phi^{-1}(P_j)$. W.l.o.g. assume that this path is a block b. Then we have

$$\begin{aligned}
c'_v(b) &= \sum_{trip_i \in b} c'_v(trip_i) + \sum_{link_i \in b} c'_v(link_i) + \sum_{other\ arc\ a \in b} c'_v(a) \\
&= \sum_{trip_i \in b} c_v(trip_i) - \sum_{trip_i \in b} \pi_i + \sum_{link_i \in b} c_v(link_i) - \sum_{link_i \in b} \pi_{i+n_t+n_d} + \sum_{other\ arc\ a \in b} c'_v(a) \\
&= c_v(b) - \sum_{a_{ij}=1} \pi_i - \sum_{a_{i+n_t+n_d j}=1} \pi_{i+n_t+n_d} \\
&= c_j - \pi P_j
\end{aligned}$$

which contradicts the assumption because no block with negative costs was found in Step 3. Now let x be an optimal solution for $P(A)$. As every feasible solution for $P(A')$ is also a feasible solution for $P(A)$, the problem $P(A')$ is bounded by cx. Especially there exists an optimal solution x' that is found by the column generation algorithm because it starts with a feasible solution. So by the first case x' is also an optimal solution for $P(A)$. □

4.2 The Subproblems

As we have seen in the previous section, an important step in the scheduling process is the efficient solution of the subproblems to generate new columns for the LP-relaxation. The subproblems appear as shortest path problems with various side conditions in the graphs.

A shortest path in the vehicle scheduling graph can be found very easily, because the vehicle scheduling graph is acyclic and therefore can be sorted topological. Moreover, a natural topological sorting is given by the time points associated with the nodes. A shortest path can then be calculated by a dynamic programming approach, that starts at the source node and check for all nodes following the topological sorting each predecessor to find the shortest path that starts at the sink. Obviously, such an algorithm has a complexity of $O(|A|)$ and can be performed rather rapidly.

The crew scheduling graph can as well be sorted topological. However, the shortest path problem turns out to be very hard because of the resource constraints. To find a feasible shortest path as defined in Definition 2.5, we have to test if for all nodes the summed consumptions of the different resources on the arcs on this path fit in the resource constraint window of the node. Since in the general formulation of the problem we can make no assumptions on the resource consumptions of the arcs, we have to collect for every node in the topological sorted list **all** paths from node n_0, unless they have the **same** combination of resource consumptions. The number of paths that have to be collected – and thereby the complexity of a shortest path algorithm – will be exponential. Therefore it is likely that every algorithm is NP-hard.

The main development of Desrochers [Desrochers 1988] is to restrict the problem in a way that still keeps all aspects of reality. They assume, that all possible resource consumptions are **discrete**, for example minutes of working time or number of breaks. Then one can store the possible paths for every **resource combination**, a number that does not grow exponentially if all windows are finite. (The problem of infinite windows is solved later by window reduction.) A shortest path can be calculated as for the vehicle scheduling graph but for every **resource combination** instead of just every node. So if on a node a working time of 6 to 8 hours with a granuality of 1 minute and one or two pieces of work are allowed, the shortest paths for 120×2 = 240 resource combinations have to be stored.

The complexity of the algorithm is

$$O(\sum_{n \in N} (|P(n)| \prod_{l=1}^{r} b_n^l - a_n^l)).$$

so to accelerate the computation and to ensure finite resource windows, a resource window reduction as in [Desrochers et.al. 1992], that cuts parts of the windows by comparison with successors and predecessors, is implemented in the current program. A detailed description of the algorithm can be found in [Desrochers 1988] and [Desrochers et.al. 1992].

5 Polyhedral Cuts

The linear relaxation of the set partitioning problem already provides very good lower bounds for the branch and bound algorithm. Nevertheless, methods that use polyhedral cuts obtain lower bounds that are higher, thereby allowing to cut more branches of the search tree, and give solutions that are "more integer". Ideally, they will generate the integer solution without a branch. In this section we will describe the overall approach of generating polyhedral cuts and explain a cutting algorithm of Hoffman and Padberg [Hoffman and Padberg 1993] for the set partitioning problem.

5.1 Set Partitioning and Polyhedral Cuts

In the formulation of the vehicle and crew scheduling problem the set of feasible solutions in the set partitioning formulation was

$$SPP(A) = conv(\{x \in I\!R^n \mid Ax = e, x \in \{0,1\}\})$$

To make use of the developments of [Hoffman and Padberg 1993], we will divide the set partitioning equalities into two sets of inequalities and get the *set packing polytope* and the *set covering polytope*:

Definition 5.1 (set packing and set covering polytope) Let A be a $n \times m$ matrix with coefficients 0 and 1, and e_m be $(1,\ldots,1) \in I\!R^m$. Then the *set packing polytope* is defined as

$$SP(A) = conv(\{x \in I\!R^n \mid Ax \leq e_m, x \in \{0,1\}\})$$

and the *set covering polytope* as

$$SC(A) = conv(\{x \in I\!R^n \mid Ax \geq e_m, x \in \{0,1\}\}).$$

□

Obviously we have $SPP(A) = SP(A) \cap SC(A)$ and so every valid inequality of $SP(A)$ or $SC(A)$ is also valid for $SPP(A)$.

The set $SPP(A)$ is a polyhedron, so it can be described by linear equalities and inequalities. Given the set of them allows to calculate the optimal solution of the VCSP by solving the linear program of minimizing cx on $SPP(A)$. Every extreme point of $SPP(A)$ is **integer** and so the optimal solution is feasible for the VCSP. Unfortunately, the inequalities are not easy to calculate and so we need some theory (see e.g. [Nemhauser and Wolsey 1988]):

Definition 5.2 (faces and facets) Let $P \subseteq \mathbb{R}^n$ be a polyhedron, and $(\pi, \pi_0) \in \mathbb{R}^n \times \mathbb{R}$ define a linear inequality $\pi x \leq \pi_0$. (π, π_0) is called a *valid inequality* for P if it is satisfied by all points in P.
Let $F = \{x \in P \mid \pi x \leq \pi_0\}$ be the intersection of the halfspace defined by (π, π_0) with P. If (π, π_0) is valid, then F is called a *face of* P. A face F is called *proper* if $F \neq \emptyset, P$.
A face F of P is called a *facet* if $\dim(F) = \dim(P)$-1. □

Obviously, F is a polyhedron that is described by certain inequalities. Every polyhedron P can be defined by the inequalities of its facets. Moreover, at least one inequality of F is necessary to describe P.

For the iteration of the algorithm we need to define special inequalities that are valid for $SPP(A)$, but cut away a solution that is feasible in the continuous relaxation of the VCSP.

Definition 5.3 (polyhedral cut) Define $S(A)$ as the continuous relaxation of the VCSP. Let x be feasible solution of $S(A)$ and (π, π_0) be a valid inequality of $SPP(A)$ with $\pi x > \pi_0$. Then (π, π_0) is called a polyhedral cut for the VCSP.

In an algorithm the polyhedral cuts (ideally facets of $SPP(A)$) are subsequently added to the problem such that the interim solutions are eliminated.

The main disadvantage of the polyhedral approach is the problem of **finding** the facets of $SPP(A)$. This is a complicated task and for NP-hard integer problems of course also NP-hard. Nevertheless it is also a great advantage to generate a subset of the facets to get a better lower bound and a solution that is more integral than the pure solution of the linear program with $S(A)$. In the next section we will give some methods to find facets for the set partitioning problem.

5.2 Intersection Graph

One way to calculate facets of a set partitioning polytope uses a graph derived from the constraint matrix as described in [Hoffman and Padberg 1993] to calculate facets of the set packing polytope $SP(A)$ and therefore for $SPP(A)$, too.

Definition 5.4 (intersection graph) Let $A = (a_{ij})_{m \times n}$ be the constraint matrix of the set partitioning problem. Then define the *intersection graph* $(\{1, \ldots, n\}, E)$ with $(j_1, j_2) \in E$ *iff* the columns a_{j_1} and a_{j_2} *intersect*, i.e., there exists an $i \in \{1, \ldots, m\}$ such that $a_{ij_1} = a_{ij_2} = 1$.
□

Definition 5.5 (cliques) Let $G = (N, E)$ be a graph and $K \subseteq N$. K is a *clique* of G *iff* it induces a maximal complete subgraph. □

These definitions allow to describe all facets of set packing type [Hoffman and Padberg 1993]:

Theorem 5.6 An inequality $\sum_{j \in K} x_j \leq 1$ defines a facet of $SP(A)$ *iff* K is the node set of a *clique* of the intersection graph. □

To detect cliques of the intersection graph, Hoffman and Padberg [Hoffman and Padberg 1993] present several algorithms. The one implemented in this work proceeds by concentrating on a subset of the column indices $F = \{j \in \{1, \ldots, n\} \mid 0 < x_j < 1\}$ that correspond to the fractionally valued variables of the current solution vector x. The subgraph of the intersection graph that is induced by F is observed to find cliques. This is done by starting with initial sets of column indices $M_r \subseteq F$ such that $a_{rj} = 1$ for all $j \in M_r$. Obviously M_r is a complete subgraph. Now the set $K \subseteq F \setminus M_r$ of columns in F that intersect with **all** columns of M_r is constructed. Any clique that contains M_r can now be build by complete enumeration on the (small) set K. Every clique $C \subseteq F$ with $\sum_{j \in C} x_j > 1$ defines a polyhedral cut that invalidates the current fractional solution. The maximal violated constraint (or a number of most violated constraints) is added to the linear program. Note that such cliques only define **facets** for the subproblem $A_F x_F = 1$ with A and x restricted to the column indices in F. A facet for the complete set partitioning problem has to be generated by a *lifting procedure* as described in [Hoffman and Padberg 1993].

In this algorithm on each step back in the branch and bound scheme the rows for the polyhedral cuts are invalidated and reinserted by the pivot and probe algorithm. This implements a *pool* of cuts as suggested by Hoffman and Padberg [Hoffman and Padberg 1993].

5.3 Polyhedral Cuts and Column Generation

The excellent effects of polyhedral cuts, especially on the depth of the branch and bound tree (see Subsection 6.2), suggest to use them in combination with the column generation algorithm of Chapter 4. Unfortunately, the cuts are additional rows of the linear program that do not correspond to single arcs of the scheduling graphs as the set partitioning and linking constraints, but to whole paths from the start to the end node. So there is no natural technique to assign the simplex multipliers to a shortest path problem.

One possibility to combine both approaches is to ignore the information provided by the simplex multipliers of the new rows and generate new columns with the set partitioning and linking

constraints. Obviously, for all implicit columns not being generated by the algorithm, their entry in all polyhedral cuts is 0. So if the simplex algorithm terminates no column with an entry other than 0 in a cutting row is the pivot column.

The main idea is to generate a shortest block or workday that is **not represented by a column already existing** and take it as a candidate for the pivot column. This approach leads to an exact algorithm that combines both polyhedral cuts and column generation. This is achieved by a branch and bound scheme similar to a k^{th} shortest path problem as in [Lawler 1972]. It calculates the current shortest path and if the corresponding column already exists, the branch step is performed as follows:

Definition 5.7 (branching on paths) Let $p = (n_1, \ldots, n_r)$ be the actual shortest path. Let

$$M = (m_1, \ldots, m_s), 1 \leq m_1 < m_s \leq r$$

be the set of indices of nodes in p that have more than one successor (usually the end of a trip with a number of possible links or a relief point with several possibilities to continue on foot). If $M = \emptyset$ no branching is possible and hence no feasible shortest path exists and the current branching node is leveled. Otherwise, s subproblems are generated (instead of two in the binary global branch and bound algorithm) as in [Lawler 1972]. Subproblem i $(1 \leq i \leq s)$ is defined by:

1. forbid arc (n_{m_i}, n_{m_i+1})
2. for all $j < i$ force arc (n_{m_j}, n_{m_j+1}) by forbidding all other arcs that start on node n_{m_j}.

The algorithm is extended by taking the value of the current shortest path as a lower bound for the desired path not coded in the LP matrix. □

To detect if the current shortest path corresponds to an existing column, all paths are coded into a set binary variables each denoting a trip or a link, respectively. This can be easily incorporated into the shortest path and column generation algorithms. The sets are hashed by adding the numbers of the arc and let the final sum modulo 512 be the index for the hashqueue. So a test of the actual path results in just a few operations in the respective hash queue and therefore is very cheap.

6 Computational Results

Now we can give some measurements of the performance of the implementation. We first describe the parameters of the instances and the different compilations of algorithms that are

used to solve them. Then the results for all combinations of examples and algorithms are presented.

6.1 Examples and Algorithms

The main characteristics of the examples are the number of trips and dtrips (the latter defined by the number of extra relief points) and the size of resource windows, especially driving and waiting times as they lead to great windows in contrast to numbers of breaks or tasks. The number of arcs for travels on foot and breaks is determined by the maximum waiting time for vehicles and crews and the number of (relief) points where the driver is *allowed* to leave his vehicle.

The problems observed in the next section have two resources, the working time and the number of breaks, and are varied by the following characteristics:

p1 3 lines, 10 trips, 16 dtrips, 38 links, 51 breaks, and 95 travels on foot. The drivers can leave their vehicle at the depot, at a central station, and at the end of every trip. The vehicle graph is composed of 42 nodes and 94 arcs, and the crew graph of 120 nodes and 391 arcs.

p1$_{rw}$ As problem p1, but with the window size reduced by 5.

p1$_{rt}$ As problem p1, but the drivers are only allowed to leave their vehicle at the depot and the central station. So the number of travels on foot is reduced to 47, and the total number of arcs in the crew graph is 319.

p2 3 lines, 20 trips, 34 dtrips, 92 links, 300 breaks, and 229 travels on foot. The drivers can leave their vehicle at the depot, at a central station, and at the end of every trip. The vehicle graph is composed of 82 nodes and 258 arcs, and the crew graph of 244 nodes and 1161 arcs.

p2$_{rw}$ As problem p2, but with the window size reduced by 5.

p2$_{rt}$ As problem p2, but the drivers are only allowed to leave their vehicle at the depot and the central station. So the number of travels on foot is reduced to 112, the number of breaks to 190, and the total number of arcs in the crew graph is 866.

p2$_{of}$ As problem p2, but all costs set to 0 except the fixed costs of vehicles and crews which are set to 1. This assignment leads to the minimization of blocks and workdays and so

P, A	cont	%	time	%	first	%	time	%	best	time
p1, a1	2747.8	100	38	100	2747.8	100	38	100	2747.8	38
$p1_{rt}$, a1	2745.6	100	26	100	2745.6	100	26	100	2745.6	26
$p1_{rw}$, a1	2582.6	98.06	9	14.06	2633.6	100	14	21.9	2633.6	64
p2, a1	4278.2	98	1433	25	4322.5	100	2564	45	4322.5	5693
$p2_{rt}$, a1	4304.7	98.8	844	5	4371.4	100.3	1712	10	4356.6	16541
$p2_{rw}$, a1	3993.1	-	1198	-	4244.4	-	3784	-	-	-
$p2_{of}$, a1	8.6	-	1335	-	11.0	-	5246	-	-	-
p3, a1	3546.6	100	2555	50	3546.6	100	4001	78	3546.6	5066
p4, a1	5716.9	-	21112	-	-	-	-	-	-	-
$p1_{rw}$, a2	2597.4	98.625	21	15.7	2649.8	100.6	45	33.6	2633.6	134

Tabelle 1: Computational results

to a feasible covering of the trips and dtrips with a minimal set. Such a problem can be solved much more efficiently with special algorithms and it is used to test the flexibility of the general algorithm presented here.

p3 2 lines, 20 trips, 40 dtrips, 86 links, 352 breaks, and 243 travels on foot. The drivers can leave their vehicle at the depot, at a central station, and at the end of every trip. The vehicle graph is composed of 82 nodes and 245 arcs, and the crew graph of 262 nodes and 1182 arcs.

p4 3 lines, 30 trips, 54 dtrips, 142 links, 616 breaks, and 374 travels on foot. The drivers can leave their vehicle at the depot, at a central station, and at the end of every trip. The vehicle graph is composed of 122 nodes and 429 arcs, and the crew graph of 374 nodes and 2007 arcs.

The composition of algorithms used as described in Chapter 4 are:

a1 Column generation, pivot and probe for all constraints.

a2 As a1, with polyhedral cuts based on clique detection.

6.2 Results

In Tables 1 and 2 the following aspects of the calculations are covered:

P, A	simplex time	%	it	cspth time	%	no. clms	no. rows	depth
p1, a1	14.8	38	885	22.8	60	321	64	0
$p1_{rt}$, a1	7.1	27	660	18.8	72	289	63	0
$p1_{rw}$, a1	43.3	67.7	3278	13.1	20.5	457	60	10
p2, a1	4715	82	36187	908	15	2268	145	3
$p2_{rt}$, a1	14960	90.4	142412	1339	8	3790	135	9
$p2_{rw}$, a1	-	-	-	-	-	-	-	>40
$p2_{of}$, a1	-	-	-	-	-	-	-	>40
$p1_{rw}$, a2	111.1	82.9	4326	14.7	10.97	443	98	3

Tabelle 2: Additional results

P, A The number of the problem with the composition of algorithms as mentioned above.

cont Value of the first continuous solution without branching. (For Algorithm a2 this is the solution **after** cut generation but **before** branching. The pure continuous solution is as for Algorithm a2.)

first Value of the first integer solution.

best Value of the best integer solution.

% Percentage, compared with the final integer solution.

time Time to compute the solution. For the best solution the time for the termination of the program and therefore the **proof** for the best solution is given.

simplex time Overall time spent for the iteration of the simplex algorithm.

it Number of iterations of revised simplex algorithm.

cspth time Overall time spent for the generation of shortest paths in the crew scheduling graph.

no. clms Number of columns at the end of the calculation.

no. rows Number of rows at the end of the calculation.

depth Length of a maximum path from the root to a leave in the branch and bound tree.

Problem	v-z*	c-z*	sum	%	time	%
p1	1254.8	$\star$	-	-	3	7
$p1_{rt}$	1254.8	$\star$	-	-	3	7
$p1_{rw}$	1254.8	$\star$	-	-	1	2
p2	1479.4	$\star$	-	-	33	1
$p2_{rt}$	1479.4	$\star$	-	-	33	1
$p2_{rw}$	1479.4	$\star$	-	-	17	-
$p2_{of}$	3.0	$\star$	-	-	68	-
p3	1177.0	2624.0	3801	107	36	1
p4	1874.8	$\star$	-	-	96	-

$\star$: no feasible solution

Tabelle 3: First vehicle then crew scheduling

As the results in Table 1 show, the limit of the current algorithm turns out to be around 20 trips. If an optimal solution is found, the lower bound by the first continuous solution is excellent (over 98 per cent of the optimal solution) and the first integer solution comes close or even is the final solution.

To calculate the first continuous solution the time used by the simplex algorithm is significantly shorter than the shortest path time (Table 2, Rows 1 and 2). If some branch steps are needed, this ratio changes to the disadvantage of the simplex algorithm because all generated columns are reused and so only few new columns have to be calculated on deeper nodes of the branch and bound tree. As all problems with depth greater than 0 show, this approach leads to a much faster generation of the lower bounds on the nodes as if all columns are erased and must be calculated again. Nevertheless, the algorithm to solve the linear programs is just a simple reversed simplex, and as for greater problems nearly 80 to 90 per cent of the time are consumed by it, a faster algorithm could improve the implementation very much.

The simultaneous application of polyhedral cuts and column generation reduces the depth of the branch and bound tree from 10 to 3 for the small instance $p1_{rw}$ (Table 2, Rows 3 and 8). At first sight this is a very promising result. As for larger instances the computation time for the generation of shortest paths in the crew scheduling graph increases drastic, no solution was obtained in reasonable time.

Tables 3 and 4 give the results for the problems above solved with a two stage algorithm:

Problem	v-z*	c-z*	sum	%	time	%
p1	1462.3	1356.5	2818.3	102	7	2
$p1_{rt}$	1555.8	1299.4	2855.2	104	10	2
$p1_{rw}$	$\star$	$\star$	-	-	-	-
p2	$\star$	2347.3	-	-	103	2
$p2_{rt}$	$\star$	$\star$	-	-	-	-
$p2_{rw}$	$\star$	$\star$	-	-	-	-
$p2_{of}$	$\star$	$\star$	-	-	-	-
p3	1801.3	2071.4	3872	109	144	3
p4	$\star$	$\star$	-	-	-	-

$\star$: no feasible solution

Tabelle 4: First crew then vehicle scheduling

vehicles before crews or crews before vehicles, respectively. Both tables present the objective function values for the single problems, the summation of the values to get the (feasible but not necessary optimal) solution of the VCSP, and the time to solve both problems independently. Value and time are compared with the optimal solution of the VCSP.

One of the main problems of the simple sequentialization used here is the fact that the second scheduling problem often has no feasible solution because of the constraints required by the solution of the first one. If the vehicles are scheduled first the resource restrictions might prevent the generation of a feasible set of workdays, e.g. that blocks are too long. This problem becomes less important in bigger examples with more possible workdays to cover the blocks. If the crews are scheduled first it is possible that the links to or from the depot are not covered, and so some vehicles have no drivers to their trips. This must be overcome by extra flow constraints on the begin and end node and is not discussed here.

Another problem that may come up if the crews are scheduled first seems to be that the branching rule is not as appropriate for the CSP as for the VCSP. In almost all cases that could not be solved the solver finds a first feasible integer solution on a step deeper than 40 in the branch and bound tree. So the tree becomes too large to be leveled in a reasonable time.

If a feasible solution is found, the average loss compared with an optimal solution to the VCSP is 5 per cent and varies from 2 to 9 per cent. The time needed to generate the result is about 2 per cent.

7 Conclusions and Further Research

In this paper we presented a mathematical formulation of the vehicle and crew scheduling problem (VCSP). It is defined as a combination of the set partitioning formulations for the vehicle scheduling problem and the crew scheduling problem with resource windows on the scheduling graph as proposed in [Desrochers and Soumis 1989]. By using this technique to model constraints for crew schedules, a great variety of different real world problems is achieved. Moreover, the VCSP presented here is easy to extend to cover multiple depot problems and additional constraints like limited numbers of vehicles or duty types (see Section 2.4).

The VCSP is solved by a column generation approach for both the vehicle and the crew scheduling part. The simplex multipliers of the set partitioning constraints for trips, dtrips, and links are subtracted from the costs of the corresponding arcs of the vehicle or crew scheduling graph, respectively. To generate the new pivot column, shortest path algorithms for both types of arcs are presented.

The column generation approach provides a good lower bound for the branching algorithm. For 10 trips the solution time is below one minute, for 20 trips above one hour, and sometimes no solution is found in an acceptable range of time. For 30 trips only the continuous relaxation could be solved. To estimate the usefulness of column generation, the branch and bound scheme was tested with all columns calculated in advance. Because of the exploding number of columns only examples containing at most 10 trips could be solved. Nevertheless the time to calculate the solution is significantly shorter without column generation.

As some examples show, the lower bound given by the continuous solution of the set partitioning problem is often not good enough to guarantee a short branching tree. To obtain lower bounds that are higher and closer to an integer solution, polyhedral cuts for the set partitioning polyhedron as described by Hoffman and Padberg [Hoffman and Padberg 1993] are incorporated in the branch and bound scheme without column generation. Even the small part of the algorithm that is implemented in this work, using only a simple clique detection mechanism and no lifting, provides lower bounds that cut the branching tree from a depth of 9 to a depth of 2.

Polyhedral cuts are combined with column generation. To satisfy that no already existing column is generated a branch and bound algorithm is developed. This algorithm increases the overall computation as much that only very small instances are solvable in reasonable time.

There are a lot of tasks which can be done in the future. As most of the computation time is required for the simplex iterations we will integrate the powerful simplex algorithm of the

state-of-the-art solver CPLEX [CPLEX]. Even if then the computation time will be shorten substantially we do not expect to solve large problems which appear in practice. Therefore to derive lower bounds we will consider graph theoretic relaxations of the set-partitioning matrix (cf. e.g. [EL-Darzi and Mitra 1995]). Additionally to obtain upper bounds truncated branch and bound algorithm will be developed.

Literatur

[Ball et.al. 1983] M. Ball, L. Bodin and R. Dial: *A Matching Based Heuristic for Scheduling Mass Transit Crews and Vehicles*, Transportation Science **17**, No. 1 (1983) 4-31.

[Bertossi et.al. 1987] A.A. Bertossi, P. Carraresi and G. Gallo: *On Some Matching Problems Arising in Vehicle Scheduling Models*, Networks **17** (1987) 271-281.

[Bodin et.al. 1983] L. Bodin, B. Golden, A, Assad, M. Ball: *Routing and Scheduling of Vehicles and Crews: The State of the Art*, Computers and Operations Research **10**, No. 2 (1983) 63-211.

[CPLEX] *Using the CPLEX Callable Library*, Version 4.0, CPLEX Optimization, Inc. (1995).

[Dell'Allmico et.al. 1993] M. Dell'Allmico, M. Fischetti and P. Toth: *Heuristic Algorithms for the Multiple Depot Vehicle Scheduling Problem*, Management Science **39**, No. 1 (1993) 115-125.

[Desrochers 1988] M. Desrochers: *An Algorithm for the Shortest Path Problem with Resource Constraints*, Cahiers du GÉRAD G-88-27, École des H.E.C., Montreal, Canada (1988).

[Desrochers et.al. 1992] M. Desrochers, J. Desrosiers, M. Solomon: *A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows*, Operations Research **40**, No. 2 (1992) 342-354.

[Desrochers et.al. 1990] M. Desrochers, J. Gilbert, M. Sauvé, F. Soumis: *CREW-OPT: Subproblem Modeling in a Column Generation Approach to Urban Crew Scheduling*, in [Desrochers and Rousseau 1990] (1990) 395-406.

[Desrochers and Rousseau 1990] M. Desrochers, J.M. Rousseau (eds): *Computer-aided Transit Scheduling*, Lecture Notes in Economic and Mathematical Systems **386**, Springer (1990).

[Desrochers and Soumis 1989] M. Desrochers and F. Soumis: *A Column Generation Approach to the Urban Transit Crew Scheduling Problem*, Transportation Science **23**, No. 1 (1989) 1-13.

[Freling et al. 1995] R. Freling, G. Boender, A. Paixão: *An Integrated Approach to Vehicle and Crew Scheduling*, Report 9503/A, Erasmus University Rotterdam, 1995.

[Hoffman and Padberg 1993] K.L. Hoffman, M. Padberg: *Solving Airline Crew Scheduling Problems by Branch-and-Cut*, Management Science **39** No.6 (1993) 657-682.

[Lasdon 1970] L.S. Lasdon: *Optimization Theory for Large Systems*, MacMillan, New York (1970).

[Lawler 1972] E.L. Lawler: *A Procedure for Computing the the K Best Solutions to Discrete Optimization Problems and its Application to the Shortest Path Problem*, Management Science **18**, No. 7 (1972) 401-405.

[EL-Darzi and Mitra 1995] E. El-Darzi, G. Mitra: *Graph Theoretic Relaxations of Set Covering and Set Partitioning Problems*, European Journal of Operational Research **87**, No. 87 (1995) 109-121.

[Nemhauser and Wolsey 1988] G.L. Nemhauser, L.A. Wolsey: *Integer and Combinatorial Optimization*, John Wiley and Sons, New York (1988).

[Patrikalakis and Xerocostas 1990] I. Patrikalakis and D. Xerocostas: *A New Decomposition Scheme of the Urban Public Transport Scheduling Problem*, in [Desrochers and Rousseau 1990] (1990) 407-425.

[Ribeiro and Soumis 1994] C.C. Ribeiro and F. Soumis: *A Column Generation Approach to the Multiple-Depot Vehicle Scheduling Problem*, Operations Research **42**, No. 1, (1994) 41-52.

[Sethi and Thompson 1984] A.P. Sethi, G.L. Thompson: *The Pivot and Probe Algorithm for solving a Linear Program*, Mathematical Programming **29** (1984) 219-233.