Planning public transport with electric buses

Han Hoogeveen, Utrecht University and Dutch Railways

Joint work with Wouter ten Bosch, Marjan van den Akker, Philip de Bruin, Jan Posthoorn, and Marcel van Kooten Niekerk

Our OR research at Utrecht University

- Our goal is to use OR to solve real-life problems; we try to include as many relevant details as possible.
- We have PhD-students financed by NS (Dutch Railways), KLM (Royal Dutch Airlines) and Qbuzz.
- Qbuzz is a bus company that provides public transport in Utrecht (among other places)
- Marcel van Kooten Niekerk is head planning of Qbuzz; he did his PhD with us (while being employed by Qbuzz).
- In this presentation I will discuss our work for Qbuzz.

Planning public transport with electric buses

How can we organize Public Transport in a reliable fashion against a reasonable price?

Contents of my talk

- Vehicle scheduling problem within Public Transport
    - diesel buses
    - electric buses (main part)
- Planning drivers (Ongoing work)
    - sequential approach
    - integrated approach

Public Transport (by bus) in the Netherlands

- The Dutch government organizes a tender for the public transport in a certain region for a period of 12 years.
- This tender specifies the conditions like the schedule that must be driven.
- Each bus company can submit a plan.
- May the best plan win ...

Busplanning in the old times

Assumptions:

- Overnight buses are stored in a single depot
- There is an unlimited number of identical buses available
- There is no limit on the length of a bus route per day (no refueling)

Input:

- Set of trips that have to be driven
- For each trip the starting place/time and end place/time are known

Goal:

- Find a feasible solution with minimum cost
- Using a bus implies a fixed cost and a variable cost per kilometer.

Analyzing the problem

- Buses can travel empty from the end point of trip i to the starting point of trip j if time permits; this is called a **deadhead**.
- A bus schedule is composed of
  - a deadhead from the depot to the starting point of its first trip
  - a set of trips and deadheads
  - a deadhead from the endpoint of its last trip to the depot.
- Trips are not important for the cost, as long as they are driven.
- We have to decide on which deadheads to drive.

Graph formulation (1)

Construct the following directed graph (direction according to time)

- Each trip i corresponds to a vertex i
- Add arc (i, j) if and only if it is possible to drive trip j immediately after trip i; this depends on the start time of j, the end time of i and the driving time from the end point of i to the start point of j at that time of day.
- There are two dummy vertices s and t corresponding to the start and end of the bus schedule at the depot
- Add arcs (s, i) and (i, t) for each vertex i

Graph formulation (2)

- A feasible bus schedule corresponds to a path from s to t in the graph.
- The goal is to find a set of paths of minimum total length that cover each trip.

Example

- Suppose that we have three trips. Trips 1 and 2 cannot be driven sequentially, but trips 1 and 3 and trips 2 and 3 can.
- Introduce vertices 1, 2, 3 and insert arcs $(1, 3)$ and $(2, 3)$. Next, add dummy vertices s and t plus (s, j) and (j, t).

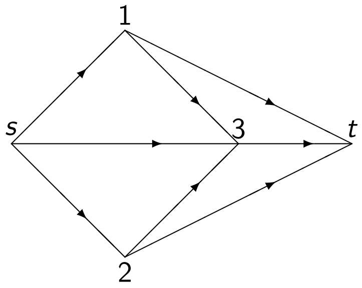


Figure: Example with 3 trips

Solution approach

Finding this set of paths can be done by either

- Solving the problem as a min cost max flow problem.
    - You must enforce that each trip is driven.
    - Thereto, split each vertex v into two vertices v_1 (start trip v) and v_2 (end trip v).
    - Add an arc (v_1, v_2).
    - Demand a flow of size 1 through (v_1, v_2).
- Solving the problem as a bipartite assignment problem; you must assign to each trip one predecessor and one successor.

Extensions

- The Multi-depot Vehicle Scheduling Problem, where we demand that the number of buses per depot remains the same over time, is solvable in polynomial time as well.
- The Multi-depot Vehicle Scheduling Problem, where buses must return to the depot they left from is $\mathcal{NP}$-hard (multi-commodity flow).
- If the buses are not identical anymore, then the problem is $\mathcal{NP}$-hard (multi-commodity flow).

New times, new challenges: Electric buses

- The battery of the electric buses does not have enough capacity to run the schedule for the diesel buses.
- The capacity depends on the temperature and hence on the season.
- To enable an efficient solution, the battery must be charged during the day.
- A solution to the e-bus scheduling problem (e-VSP) describes for each bus the trips that must be driven (including deadheads) together with a schedule that describes when to charge (at a suitable place).
- Marcel van Kooten Niekerk considered this problem (among others) in his PhD thesis; this paper has appeared in the journal Public Transport.

Charging and discharging characteristics

- The current charge of a battery is specified as a percentage: 100% is full, and 0% is empty.
- The Depth of Discharge (DoD) is defined as the percentage by which the battery is discharged at the most.
- Charging a typical battery (Li-ion) is not linear in time:
    - From 0% to 80% the amount of charge is linear with the amount of time spent.
    - After 80% the process slows down.
    - From 0-80 takes as much time as from 80-100.

Lifetime of a battery

- The lifetime of a battery is specified in the Cycle Lifetime: the number of times a battery can be discharged for 100%.
- The lifetime is further determined by the Depth of Discharge (DoD).
- The lifetime is not linear in the DoD: 10 times 10% is much better than 1 time 100%. **Try to reduce the DoD**.
- Using regression, for a typical battery Marcel estimated the function

$$cycles(x) = 4825,3e^{2,519x},$$

  where x is the DoD value and $cycles(x)$ is the number of charge/discharge cyles that can be made until end-of-life.
- The wear-out cost of the battery can be approximated using this function.

Charging

- Charging must take place at a charging station
- We assume that the infrastructure is given
- We assume that there is enough capacity to charge multiple buses at the same time.

Cost of a charging schedule

- The charging schedule determines the wear-out cost (batteries are very expensive!)
- The price of the electricity varies over the day; this can be taken into account (not yet done; later more about this).

Cost of a charging schedule

- The charging schedule determines the wear-out cost (batteries are very expensive!)
- The price of the electricity varies over the day; this can be taken into account (not yet done; later more about this).

Important consideration

It is better for the lifetime to have a small DoD, but for efficiency it is better to have a larger DoD.

Additional features of the e-VSP

- The basis stays the same: trips, deadheads, etc.
- Driving a trip or deadhead requires a given amount of charge.
- Buses must charge to avoid an empty battery; this can happen in between trips, which may require another deadhead.
- The time required for charging depends on the current charge and the target charge.
- The cost of driving is determined by the cost of the charging schedule.

Our previous work

In his PhD-thesis Marcel has presented three models with the following properties.

Property	Model 1	Model 2	Model 3
	continuous	discrete	CG
Charge variable	Exact	Rounded	Rounded
Time-of-day price electricity	No	Yes	Yes
Non-linearity of charging time	No	Yes	Yes
Effects DoD on lifetime	No	Yes	Yes
Maximum problem size	$\leq$ medium	$\leq$ medium	Large
Optimal solution guaranteed	Yes	Yes	No

I will highlight the ideas behind the Column Generation approach.

Column generation approach

- A bus schedule is meant to be driven by one bus; it contains the list of trips that the bus drives together with a charging scheme.
- A solution to the e-VSP consists of a set of feasible bus schedules that together cover all trips exactly once.
- Suppose that we know all possible, feasible bus schedules; then we can find the best combination by solving the problem as an Integer Linear Program.

Notation

- T: set of trips that must be driven (remark that a trip of the same line at another time is another trip).
- S: set containing all feasible bus schedules (charging included).
- c_s: cost of bus schedule s
- a_{ts}: parameter that indicates whether trip t ($t \in T$) is included in bus schedule s ($s \in S$).

$$a_{ts} = \begin{cases} 1 & \text{if } s \text{ contains trip } t \\ 0 & \text{otherwise} \end{cases}$$

- Introduce for each bus schedule s a binary variable x_s that indicates whether bus schedule s is chosen.

ILP formulation

$$\min \sum_{s \in S} c_s x_s$$

$$\sum_{s \in S} a_{ts} x_s = 1 \quad \forall t \in T \qquad (\text{ or } \geq 1)$$

$$x_s \in \{0, 1\} \quad \forall s \in S$$

The first constraint ensures that each trip is covered once in the solution.

ILP formulation

$$\min \sum_{s \in S} c_s x_s$$

$$\sum_{s \in S} a_{ts} x_s = 1 \quad \forall t \in T \quad (\text{ or } \geq 1)$$

$$x_s \in \{0, 1\} \quad \forall s \in S$$

The first constraint ensures that each trip is covered once in the solution.

Complication

We do not know all feasible bus schedules, and the overwhelming majority of bus schedules is useless.

Finding a good subset of S (1)

- We only need the bus schedules that are needed to find a very good (preferrably optimal) solution.
- Therefore, we try to find a subset of S that we hope contains these desired bus schedules.

Basic idea

Bus schedules that are needed to solve a relaxation of the original problem (ILP) might be useful to solve the original problem as well.

Finding a good subset of S (2)

- The relaxation that we look at is the LP-relaxation: relax the constraints $x_s \in \{0, 1\}$ to $x_s \geq 0$ (the other constraints prevent x_s from exceeding 1).
- We solve the LP-relaxation using Column Generation.
- This approach together with some additional features has been proven very successful in solving for example the Gate Assignment problem and Nurse Rostering problems.

Column generation

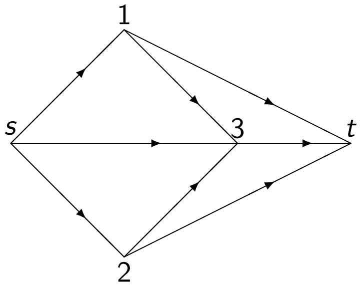
Approach in a nut shell:

1. Start with any subset of bus schedules that allows a feasible solution to the LP
2. Solve the LP for the current set of bus schedules
3. Check whether you can improve the current solution by adding a new feasible bus schedule to the current subset of bus schedules; this is done by solving the **pricing problem**.
4. If you can find such a bus schedule, then add it and continue; otherwise, stop.

Pricing problem

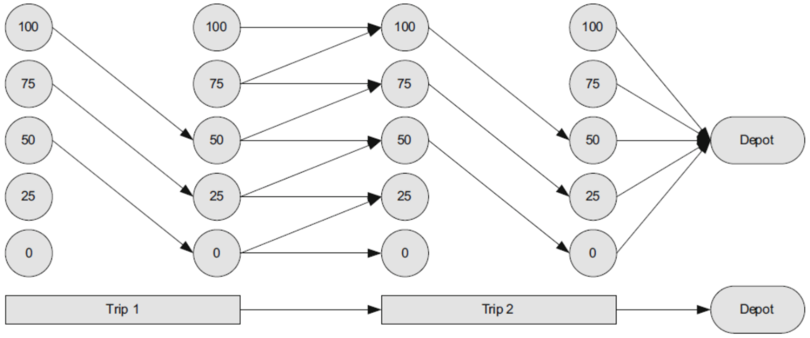
- Goal is to find a feasible bus schedule that improves the current LP-solution
- The cost should be as small as possible, but you get a reward for driving trips; this reward depends on the current LP-solution (shadow prices, etc.).
- Remark that a bus schedule corresponds to a path in the graph, just like for the problem with the diesel buses.
- The battery constraint and charging can be included by adding nodes and arcs (PhD thesis by Marcel).
- To solve the pricing problem we must solve a shortest path problem in an extended graph.

Example continued



- Split a trip vertex in a ‘start’ and ‘end’ point with an arc in between.
- Discretize the current charge (for example 0%, 25%, 50%, 75%, 100%) at that point.
- Include arcs between charge levels to depict the charge possiblities.

Graph for solving the pricing problem



The cost of a trip is modified by including the shadow prices!

Options to find a feasible integral solution

- If you can solve the LP-relaxation to optimality, then you can apply branch-and-price to find an optimal solution (may require quite some time; not applicable here).
- Solve the ILP for a subset of all feasible bus schedules. You can use the bus schedules discovered when solving the LP-relaxation plus a set of additional schedules generated by perturbations, etc. This usually works very well, **but not in this case.**
- Use some rounding heuristic. For example, choose one bus schedule, update the instance, and solve the LP-relaxation and ILP again.

Currently not as successful as hoped for

We had to use the latter option, but this increases the cost more than desired. The gap between the lower bound (LP-relaxation) and the solution found was more than 1%.

Can we do better?

Are we to blame, or is the gap due to a bad lower bound?

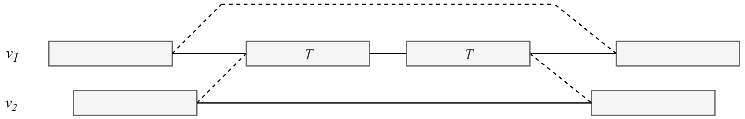
Local search

‘If in need of a good solution, then you can always try local search’.

- Simulated annealing approach.
- Start with an initial solution: let every trip be driven by a separate bus.
- Two neighborhoods (equal probability):
  - Move a trip (next slide)
  - Swap tails (next slide)
- We only consider feasible solutions.
- We charge whenever possible. The price of electricity is not taken into account yet; this can be easily included (will be discussed later).
- We use a multistart strategy.

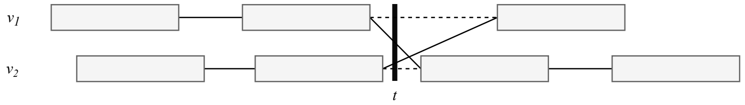
Move neighborhood

- Choose two vehicle schedules v_1 and v_2
- Select a range of trips to be moved from v_1 to v_2



Swap neighborhood

- Choose two vehicle schedules v_1 and v_2
- Select a point in time t
- Swap the tails of the bus schedules v_1 and v_2 after time t



Cost of charging

Situation

You know the schedule of a bus => you know the possible charging intervals.

- Electricity prices fluctuate tremendously. It is easy to make general predictions of the cost per hour.
- Use these predictions to find the charging schedule with minimum **expected** price. This includes the wear out cost of the battery.

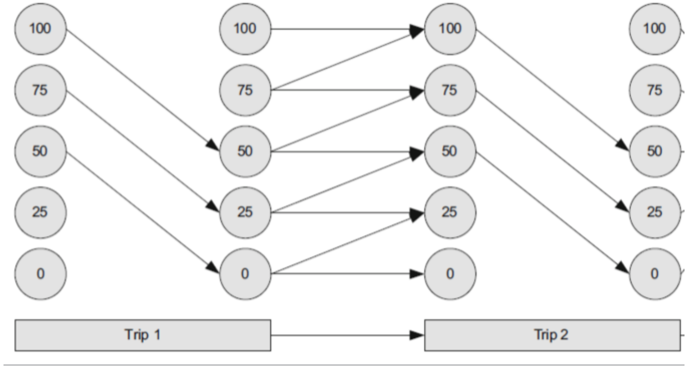
Finding the best charging schedule (1)

- We can solve this problem as a shortest path problem with **discretized charging** levels (example will come up soon).
- Dummies s and t correspond to charging during the night at the depot; the corresponding charge is 100%.
- An optimal charge schedule corresponds to a shortest path from s to t.

Finding the best charging schedule (2)

- Introduce a cluster of vertices corresponding to charge levels at the **start and at the end** of each possible charging interval *CI*.
- Connect the charge vertices at the start with the charge vertices at the end of *CI* with arcs that correspond to charging the battery with some feasible amount.
- Connect the charge vertices at the end of a *CI* with the charge vertices at the end of the next *CI* to account for the charge used for driving.
- Since we know the corresponding charge levels and time of charging, the cost of each arc is readily determined.

Finding the best charging schedule: example



Simulated Annealing with ILP

- The multistart Simulated Annealing produces decent results ...

Simulated Annealing with ILP

- The multistart Simulated Annealing produces decent results ...

and it produces loads of feasible bus schedules.

Simulated Annealing with ILP

- The multistart Simulated Annealing produces decent results ...

and it produces loads of feasible bus schedules.

- We can use these to form our set S that we need to solve the ILP.

As far as we know, this is a new solution approach.

Simulated Annealing with ILP

- The multistart Simulated Annealing produces decent results ...

and it produces loads of feasible bus schedules.

- We can use these to form our set S that we need to solve the ILP.

As far as we know, this is a new solution approach.

~~This is a new solution approach~~

- Unfortunately for us, it is not new.
- Fortunately for you and Qbuzz: it works very well!

ILP formulation

$$\min \sum_{s \in S} c_s x_s$$

$$\sum_{s \in S} a_{ts} x_s = 1 \quad \forall t \in T$$

$$x_s \in \{0, 1\} \quad \forall s \in S$$

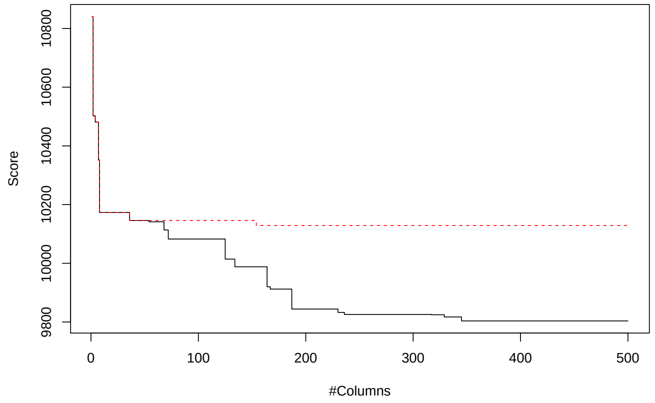
Results

Dataset	CG-LP	CG-ILP	Gap	SA-M	Gap	SA-ILP	Gap
3	23920	24322	1.68%	24018	0.41%	23920	0.00%
7	20712	21568	4.13%	21018	1.48%	21004	1.41%
8	21446	22020	2.68%	21732	1.33%	21530	0.39%
7+8	42344	time	-	42764	0.99%	42394	0.12%
3+7+8	66183	time	-	67610	2.15%	66878	1.05%

Headers:

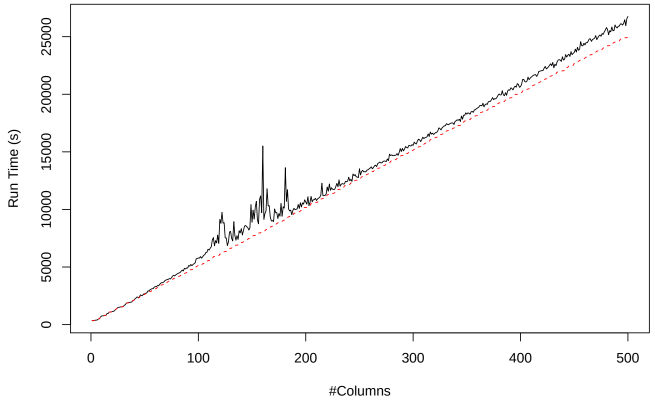
- CG-LP is the value of the LP-relaxation;
- CG-ILP is the solution of the ILP with S equal to the bus schedules discovered by the Column Generation;
- SA-M: optimal solution of the multistart SA over 100 runs.
- SA-ILP: result of the ILP with input the bus schedules found by SA-M

Effect of the number of SA runs on the quality



The red dots indicates the best value found in the runs of SA; the black line indicates the value found by the ILP.

Effect of the number of SA runs on the runtime



The red dots indicates the amount of time needed for the runs of SA; the black line indicates the total amount of time needed.

Summarizing the first part

- We solve the problem by combining individual bus schedules to a solution for the e-VSP
- We compared two techniques to determine the individual bus schedules:
  - Column Generation: we find one bus schedule at a time, but we take the 'combinability' into account by solving the pricing problem
  - Simulated Annealing: here we look for full solutions to the e-VSP, which are then split.
- In contrast to earlier experiences, Column Generation did not work very well; Simulated Annealing saved the day.
- Qbuzz will use these results to plan the buses in Utrecht (total cost 200M per year).

Research questions left: explaining the (lack of) success

Why it is so successful?

- Why did Column Generation fail? Can we identify reasons why the bus schedules needed to solve the LP-relaxation are not useful to solve the ILP?
- Does Simulated Annealing always find good columns, or it is just luck? How about extending neighborhoods? It may work contraproductive.

Observations

- You need a lot of variation in the bus schedules produced by local search.
- A lousy implementation of Simulated Annealing is perfect for that purpose.

Research questions left: applying it to other problems

- Simulated Annealing is a very general technique, just like ILP.
- Using the combination of the two is much easier than using Column Generation
- A disadvantage of Column Generation is that you need to solve the pricing problem; if you do not find an improvement to the current solution, then the method will stop.
- Which problems are suitable to this approach? The potential seems huge; we have applied it successfully already a number of times.

Planning the drivers: sequential approach

- Major assumption: **we have decided on the bus schedule already!**
- We need to plan the drivers given the trips and deadheads that must be driven; we need a driver for each shift (or at least one).
- Drivers can work in several shifts: morning, day, evening, broken shift.
- Individual duties (rosters) must comply with working regulations: minimum working hours, maximum working hours, short breaks, meal breaks, etc.
- Moreover, we want to have robust solutions (can also be used as a pee break).
- Given a duty, we can easily determine its cost.

ILP approach

- We can formulate it as an ILP using **duties** (rosters)
- Given the entire set of feasible rosters, we must select a subset with minimum total cost such that all trips and deadheads are covered.
- This ILP is 'the same' as the one for the vehicle scheduling problem.

Notation for the duty selection problem

- D: set of all feasible duties.
- c_d: cost of duty d
- b_{td} and b_{qd} indicate whether trip t and deadhead q are included in duty d
- y_d is a binary decision variable for each duty $d \in D$ that indicates whether duty d is chosen.
- T and Q are the sets containing all trips and **necessary** deadheads.

ILP formulation

$$\min \sum_{s \in S} c_d y_d$$

$$\sum_{d \in D} b_{td} y_d = 1 \quad \forall t \in T \quad (\text{ or } \geq 1)$$

$$\sum_{d \in D} b_{qd} y_d = 1 \quad \forall q \in Q \quad (\text{ or } \geq 1)$$

$$y_d \in \{0, 1\} \quad \forall d \in D$$

Again: we do not know all feasible duties, and the overwhelming majority of bus schedules is not necessary.

Branch-and-price approach

- We have tried to solve this problem to optimality using branch-and-price.
- First step: find a very good heuristic solution by applying the **combination of local search and ILP** approach (similar to the one we used for the bus scheduling problem).
- Second step: solve the LP-relaxation to optimality using column generation.

Solving the LP-relaxation to optimality

- The pricing problem corresponds to a **constrained shortest path problem**; here we take the **working regulations** into account.
- The vertices in the graph correspond to the start/end of a trip or deadhead; hence we know the corresponding time at which this event takes place.
- There are arcs between the vertices other than the ones corresponding to trips and deadheads; these correspond to simply waiting at the spot. If desired, you can also add taxi rides.
- If we know the start and end vertex of the duty, then we know the type of shift, number of breaks needed, etc.
- This problem is solved using a **labeling algorithm** (comparable to dynamic programming).

Branching rules

- Our first type of branching rule is based on the number of shifts of a certain type.
- This part is successful: in this way we can determine how many shifts of each type we should use.
- Next, we can branch on using deadheads, but this has a limited success.
- If an optimal solution is not determined quickly, then it is not possible (for us at least) to find an optimal solution.

If we cannot find an optimal solution, then we should be satisfied with a very good solution.

We can again use our combination of Simulated Annealing and ILP to find a good solution; we can use here our knowledge about the types of shifts needed.

Solving the integrated problem

Observations

- We do not know which deadheads are required, and there are very many possible deadheads.
- Local search is hard to apply on the integrated problem because of the dependencies: if you plan a deadhead, then you need both a bus and a driver.

- There are two different subproblems: e-VSP and duty scheduling problem.
- Main question: how can we enforce that the solutions to the two subproblems are compatible?

ILP formulation

- ILP formulation using possible **bus schedules** and possible **duties**.
- Select bus schedules and duties such that
  - All trips are covered by a bus and a driver.
  - Possible deadheads **can be selected if and only if there is a driver**!

ILP formulation

- ILP formulation using possible **bus schedules** and possible **duties**.
- Select bus schedules and duties such that
  - All trips are covered by a bus and a driver.
  - Possible deadheads **can be selected if and only if there is a driver**!

Complication

There are very many possible deadheads => MANY constraints

Notation: ILP for the integrated approach

- Trips: T is the set of all trips (index t);
- Deadheads: Q is the set of all possible deadheads (index q).

Notation: ILP for the integrated approach

- Trips: T is the set of all trips (index t);
- Deadheads: Q is the set of all possible deadheads (index q).
- Bus schedules:
    - S: set of all bus schedules;
    - c_s: cost of bus schedule s
    - a_{ts} and a_{qs} indicate whether trip t and deadhead q are included in bus schedule s
    - x_s is a binary decision variable for each bus schedule $s \in S$

Notation: ILP for the integrated approach

- Trips: T is the set of all trips (index t);
- Deadheads: Q is the set of all possible deadheads (index q).
- Bus schedules:
    - S: set of all bus schedules;
    - c_s: cost of bus schedule s
    - a_{ts} and a_{qs} indicate whether trip t and deadhead q are included in bus schedule s
    - x_s is a binary decision variable for each bus schedule $s \in S$
- Duties:
    - D: set of all feasible duties.
    - c_d: cost of duty d
    - b_{td} and b_{qd} indicate whether trip t and deadhead q are included in duty d
    - y_d is a binary decision variable for each duty $d \in D$

ILP formulation integrated approach

$$\min \sum_{s \in S} c_s x_s + \sum_{d \in D} c_d y_d$$

$$\sum_{s \in S} a_{ts} x_s = 1 \quad \forall t \in T \qquad \text{(drive all trips)}$$

$$\sum_{d \in D} b_{td} y_d = 1 \quad \forall t \in T \qquad \text{(a driver for each trip)}$$

$$\sum_{s \in S} a_{qs} x_s = \sum_{d \in D} b_{qd} y_d \quad \forall q \in Q \qquad \text{(a driver for each chosen deadhead)}$$

$$x_s, y_d \in \{0, 1\} \quad \forall s \in S, d \in D$$

Decoupling the subproblems

What makes it complicated

- We can solve the e-VSP problem and the duty scheduling problem independently.
- For a given set of bus schedules and duties, we can solve the combined ILP.
- The problem is find bus schedules that enable using good duties, and vice-versa.

Remedy

- We decouple the two subproblems by removing the deadhead constraints, which connect the two.
- Hereto, we apply Lagrangean relaxation to the deadhead constraints.

Lagrangean problem

- Introduce Lagrangean multipliers λ_q for all $q \in Q$
- For all $q \in Q$:
  - Remove constraint q

    $$\sum_{s \in S} a_{qs} x_s = \sum_{d \in D} b_{qd} y_d$$

    from the set of constraints.
  - Add the term

    $$\lambda_q \left(\sum_{s \in S} a_{qs} x_s - \sum_{d \in D} b_{qd} y_d \right)$$

    to the objective function.

Resulting problem

$$\min \sum_{s\in S}(c_s + \sum_{q\in Q} \lambda_q a_{qs})x_s + \sum_{d\in D}(c_d - \sum_{q\in Q} \lambda_q b_{qd})y_d$$

$$\sum_{s\in S} a_{ts}x_s = 1 \quad \forall t \in T$$

$$\sum_{d\in D} b_{td}y_d = 1 \quad \forall t \in T$$

$$x_s, y_d \in \{0,1\} \quad \forall s \in S, d \in D$$

Lagrangean relaxation

- Since the e-VSP and duty scheduling problem are no longer connected, we can split this problem into two separate, independent problems with Lagrangean multipliers in the cost
- The cost of using deadhead q in the e-VSP is increased by λ_q, since the cost of bus schedule s has now become

$$(c_s + \sum_{q \in Q} \lambda_q a_{qs})$$

- Similarly, the cost of using deadhead q in the e-VSP is decreased by λ_q in the duty scheduling problem, since cost of duty d has now become

$$(c_d - \sum_{q \in Q} \lambda_q b_{qd})$$

How further?

Basic iteration

- We solve the bus scheduling problem using local search.
- We solve the LP-relaxation of the duty problem by column generation. This is **independent from** the solution to the e-VSP! It is also possible to apply local search here.
- Store the discovered bus schedules and duties.

How further?

Basic iteration

- We solve the bus scheduling problem using local search.
- We solve the LP-relaxation of the duty problem by column generation. This is **independent from** the solution to the e-VSP! It is also possible to apply local search here.
- Store the discovered bus schedules and duties.

- Adjust the Lagrangean multipliers to improve the coordination between the solutions of the subproblems.
- If there is still time left, then apply another iteration.
- Finally, solve the ILP with the discovered bus schedules and duties.

Adjusting the multipliers

- The two subproblems seem to be independent, but they are both influenced by the Lagrangean multipliers.
- Compare the solutions to the subproblems:
  - If there is a deadhead q in the e-VSP solution without a driver, then decrease λ_q: this makes the deadhead more expensive for the e-VSP and cheaper to cover for the duty scheduling problem.
  - If there is a driver for deadhead q, but this deadhead is not in the e-VSP solution, then increase λ_q: this makes the deadhead cheaper for the e-VSP and more expensive for the duty scheduling problem.

By adjusting the Lagrangean multipliers, we hope that the two subproblems converge to a feasible solution for the integrated problem!

Results

- Ongoing work
- Reasonable computation times
- You can stop generating columns when you are running out of time
- Preliminary computational experiments: gain of 5%-10% compared to the sequential approach.

Wrap-up

Decomposition for e-VSP

- We compared two techniques to determine the individual bus schedules:
    - Pricing within Column Generation
    - Split up solutions found with Simulated Annealing
- ILP fed by Simulated Annealing was great.
- Qbuzz will use these results to plan the buses in Utrecht (total cost 200M per year).

Wrap-up

Decomposition for e-VSP

- We compared two techniques to determine the individual bus schedules:
  - Pricing within Column Generation
  - Split up solutions found with Simulated Annealing
- ILP fed by Simulated Annealing was great.
- Qbuzz will use these results to plan the buses in Utrecht (total cost 200M per year).

Ongoing work: planning drivers

- The integrated approach combines Lagrangean relaxation, column generation, and local search.
- Preliminary results are very promising. Gain of 5%-10% compared to the sequential approach.

That's it

Questions?

Han Hoogeveen
Utrecht University
j.a.hoogeveen@uu.nl