Task Discussion
Carsharing

Daniel Graf

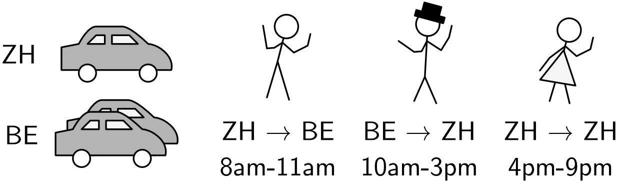
ETH Zürich

December 16, 2015

This task is based on the paper "*Scheduling Transfers of Resources over Time: Towards Car-Sharing with Flexible Drop-Offs*" by Kateřina Böhmová, Yann Disser, Matúš Mihalák and Rastislav Šrámek.

Problem Statement

- **Given:**
  - A list of S rental stations with the number of initally available cars.
  - A list of N car rental requests each with start and end time and location and its profit.
- **Wanted:** Find a feasible subset of the requests that maximizes the total profit.
- **Subtasks:**
  1. Only a single car
  2. At most 20 requests
  3. At most 10000 requests
  4. Many different times
  5. Many rental stations



First Subtask: A Single Car

For a single car and only two stations, we can efficiently solve all the subproblems of the following form using **Dynamic Programming**:

> *What is the maximum profit if the car ends at stations s at time t?*

Such a DP-table `DP[s][t]` has size $2 \times \lceil 100'000/30 \rceil$ and can be filled quickly:

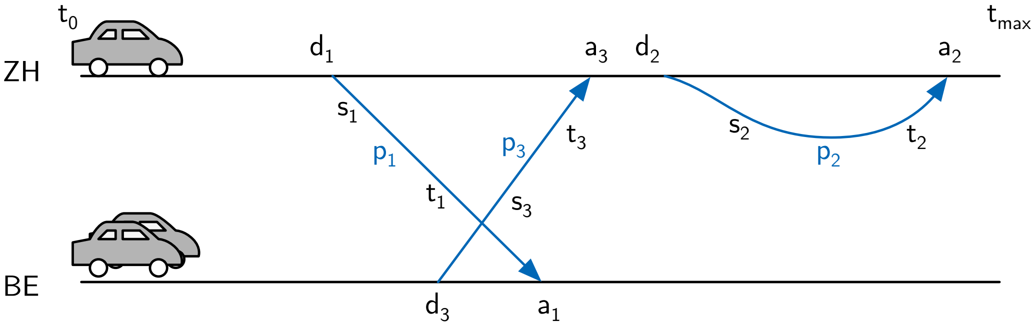
- initialize the cells with `DP[s][0] = 0` and `DP[s][t] = DP[s][t-1]`
- for all requests `r` that end at time $30 \cdot t$ at stations s:
  `DP[s][t] = max(DP[s][t], DP[r.s][r.d/30] + r.p);`
- final answer is `max(DP[0][MAXT-1], DP[1][MAXT-1])`

This solution does not easily generalize to multiple vehicles.

Second Subtask: Min Cost Max Flow

Instead, let us model it as a flow problem:

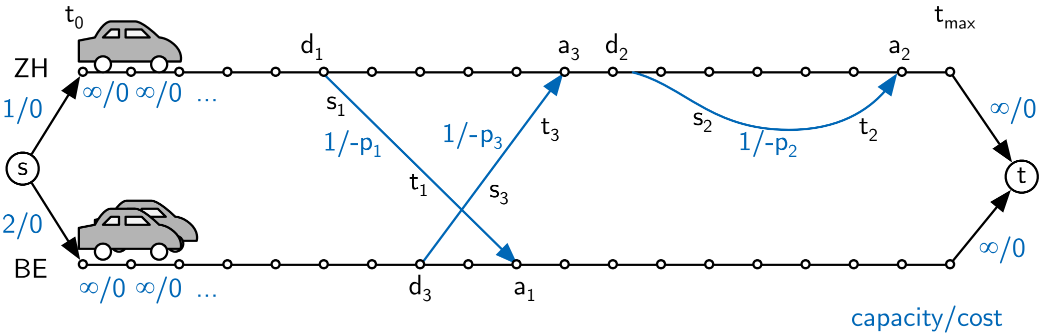
Every car is a unit of flow that flows through time and space.



Second Subtask: Min Cost Max Flow

Instead, let us model it as a flow problem:

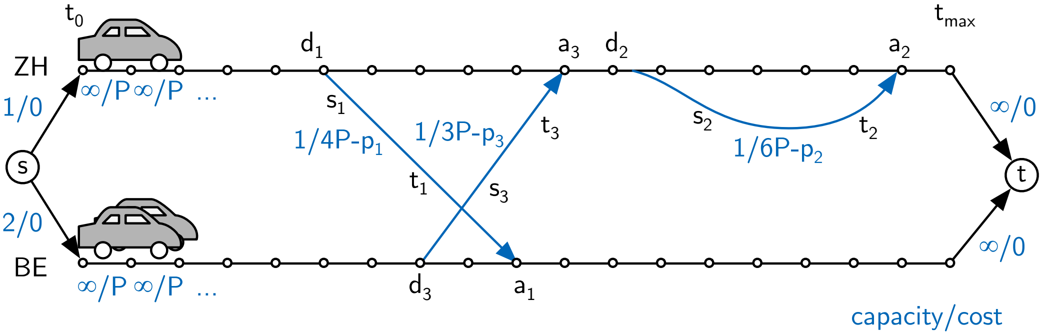
Every car is a unit of flow that flows through time and space.



Use `cycle_canceling` to handle the negative weigths.

Third Subtask: Handle more requests

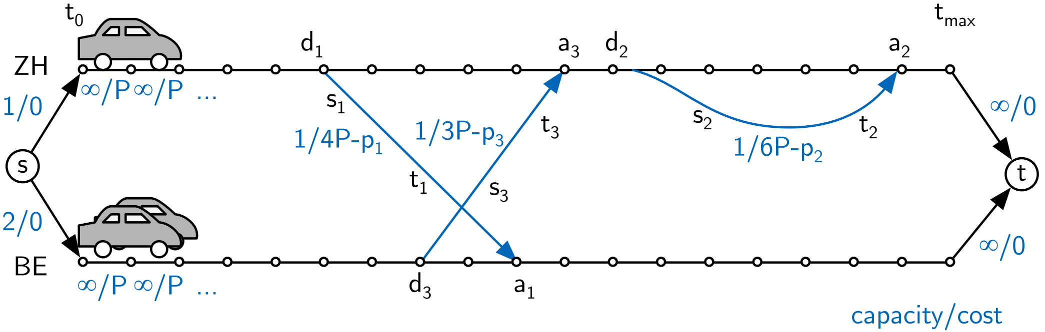
The usual speedup problem: transform the graph to get non-negative weights so that we can use the faster Min Cost Max Flow algorithm.



Add offset P (= the maximum profit per request) to every timestep.
Note that every s-t-path encounters the same amount of P offsets.

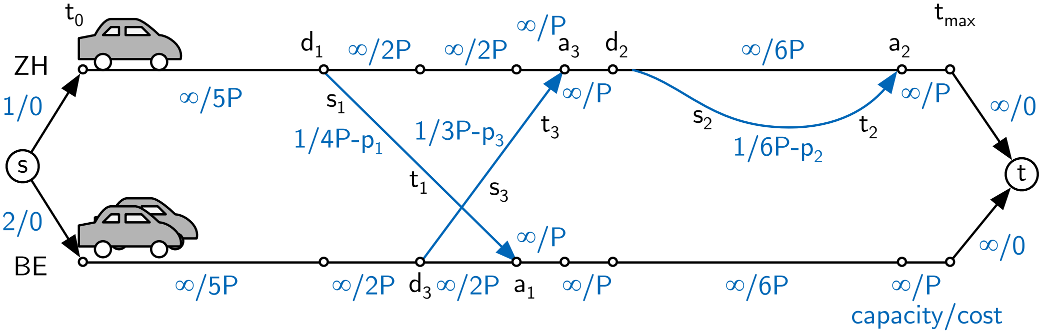
Fourth Subtask: Fine-grained time resolution

If the time resolution is 1 instead of 30 minutes, the graph gets 30 times bigger.
Note: At most $20'000$ out of $200'000$ vertices have degree > 2 at all.
We can compress these long paths and only create vertices for timesteps where some request starts or ends. This technique is often called *coordinate compression* and reduces the graph size roughly 10 times.



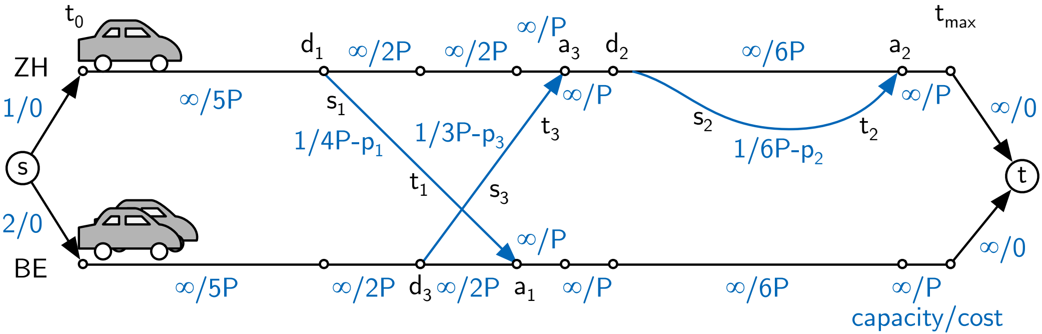
Fourth Subtask: Fine-grained time resolution

If the time resolution is 1 instead of 30 minutes, the graph gets 30 times bigger.
Note: At most $20'000$ out of $200'000$ vertices have degree > 2 at all.
We can compress these long paths and only create vertices for timesteps where some request starts or ends. This technique is often called *coordinate compression* and reduces the graph size roughly 10 times.



Fifth Subtask: Multiple Stations

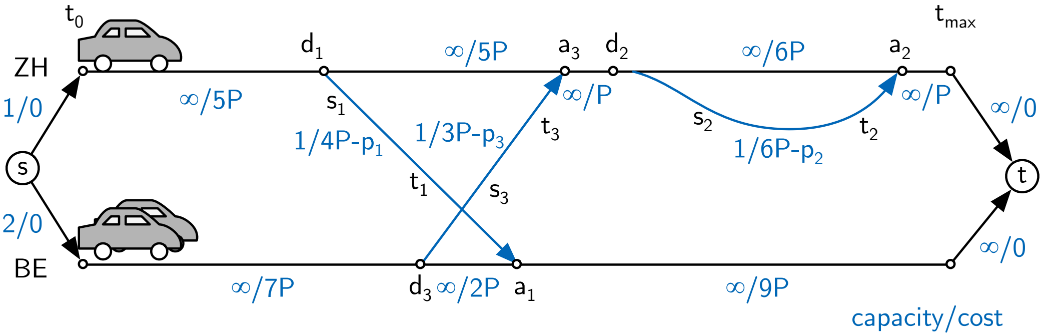
Compressing the coordinates per station reduces the graph size again S times.



Implementation detail: Use a `std::map<int,int>` per station to easily map the input times to their compressed coordinates and get the neighboring timestamps.

Fifth Subtask: Multiple Stations

Compressing the coordinates per station reduces the graph size again S times.



Implementation detail: Use a `std::map<int,int>` per station to easily map the input times to their compressed coordinates and get the neighboring timestamps.