Efficient Route Planning

SS 2011

Lecture 10, Friday July 22nd, 2011
(Transit networks again, multi-label Dijkstra)

Prof. Dr. Hannah Bast
Chair of Algorithms and Data Structures
Department of Computer Science
University of Freiburg

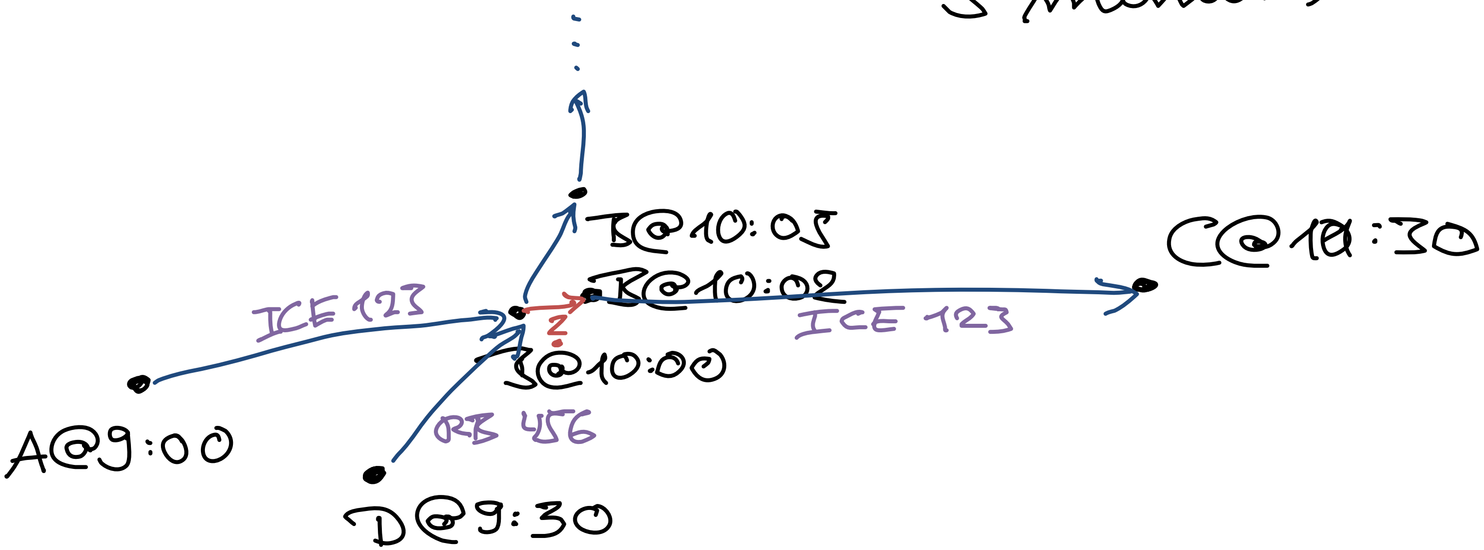
Overview of this lecture

- Organizational
  - Your results from Ex. Sheet #6 (transit node routing)
  - This is the **third to last** lecture
- Transit Networks Reloaded
  - Transfer buffers in the time-dependent model
  - Details about the arcs between arrival, departure and transfer nodes in the time-expanded model
- Multi-criteria costs
  - How to model → Pareto sets
  - How to compute shortest paths → Multi-label Dijkstra
- Exercise sheet
  - You get one more week for Exercise Sheet #7

Transfer buffers 1/5

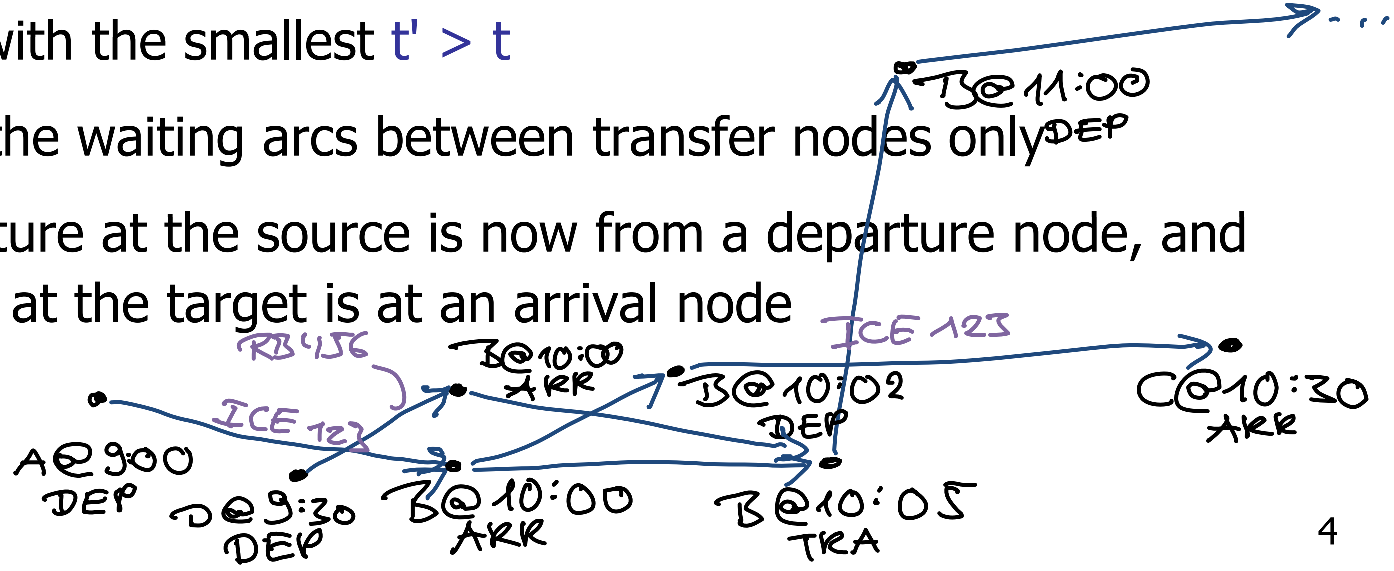
- Time-expanded model
  - This is non-trivial, because we need to distinguish between staying on a vehicle at a station (which must not require any transfer time) and changing the vehicle, for example:

Transfer buffer
5 minutes



Transfer buffers 2/5

- Time-expanded model, solution
  - Split up each node from before into an **arrival node** and a **departure node**, and add an arc between the two (we can also model layover time that way now)
  - For each arrival node A@t, add a **transfer node** A@t' and an arc from A@t to A@t', where t' – t is the transfer buffer
  - For each transfer node A@t, add an arc to the departure node A@t' with the smallest t' > t
  - Have the waiting arcs between transfer nodes only
  - Departure at the source is now from a departure node, and arrival at the target is at an arrival node



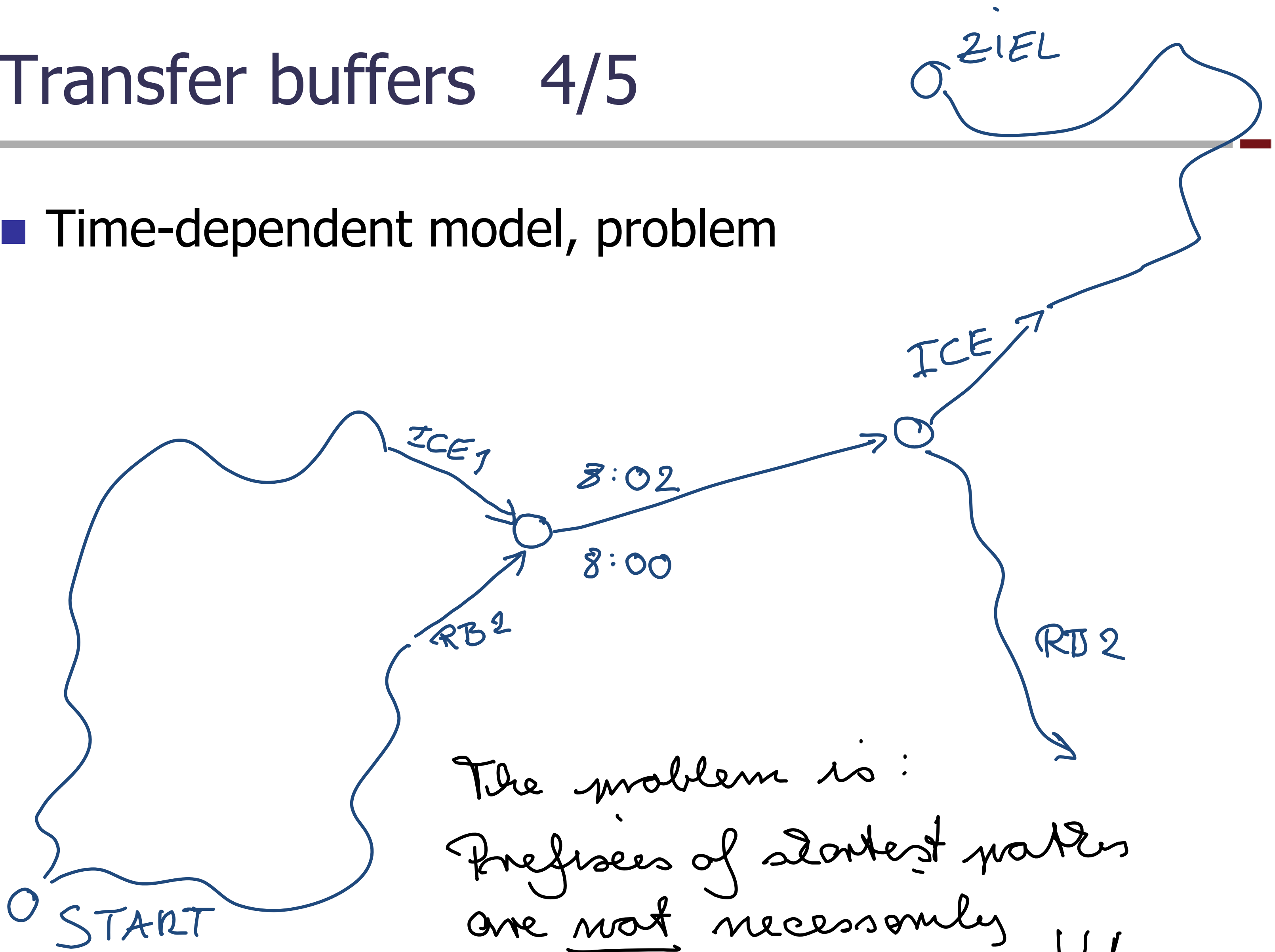
Transfer buffers 3/5

- Time-dependent model, solution 1
  - We also have to distinguish here between staying on a vehicle and changing the vehicle at a station
  - It looks like we can do this by simply remembering for each node, along with the tentative arrival time $t[u]$, the id ℓ of the vehicle with which we arrive at u
  - Then we can build the transfer buffer into the cost function

    $\mathrm{cost}_{u,v}(t, \ell)$ = time to reach v, if we are at u at time t sitting in vehicle ℓ
  - Unfortunately, Dijkstra's algorithm will not always correctly compute the shortest path anymore then ... why?

Transfer buffers 4/5

- Time-dependent model, problem



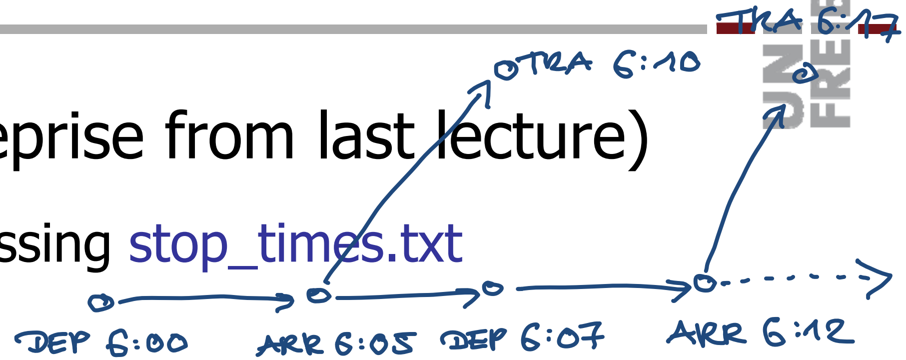
The problem is:
Prefixes of shortest paths are not necessarily shortest paths anymore !!!

Transfer buffers 5/5

- Time-dependent model, solution 2
  - Have separate arrival and departure nodes, too
  - One arrival and one departure node per line suffices
  - But we no longer only have one node per station then
- Time-dependent model, solution 3
  - When we can arrive at a station at two different times t_1 and t_2 with different vehicles, and $|t_2 - t_1|$ is $\leq$ the transfer buffer, pursue both possibilities
  - Then we need to do a multi-label Dijkstra (Dijkstra maintaining several shortest paths to the same node), see second half of this lecture

Arrival, departure, transfer nodes 1/4

- Step 1: Parse from GTFS (reprise from last lecture)
  - Create **all** nodes while processing stop_times.txt
  - And also the following arcs:
    - between arrival and departure nodes ("traveling arcs")
    - from arrival nodes to transfer nodes ("alighting arcs")



Arrival, departure, transfer nodes 2/4

- Step 1: Parse from GTFS , continued ...
  - While processing stop_times.txt, also maintain for each station the list of departure and transfer nodes of that station, with their time and type (departure or transfer)

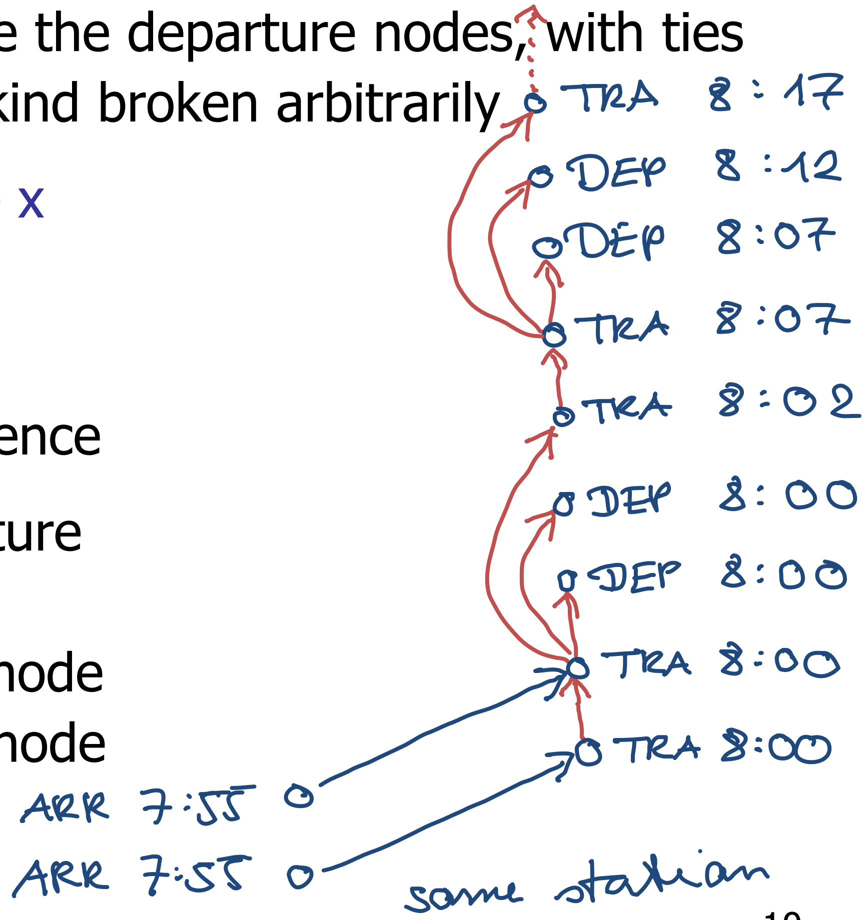
    std::vector<std::vector<Node> > _nodesPerStation;

    Note: in GTFS the stations are strings, but it's more efficient to convert them into consecutive station ids during the parsing of stops.txt; remember the correspondence like this:

    hash_map<std::string, int> _stationIdPerName;
  - It remains to add the following arcs:
    - from transfer nodes to departure nodes ("boarding arcs")
    - from one transfer node to the next ("waiting arcs")

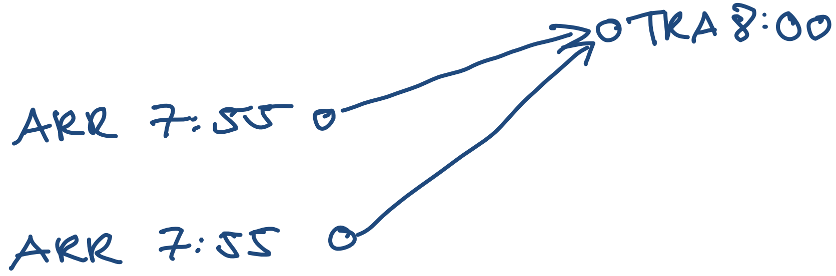
Arrival, departure, transfer nodes 3/4

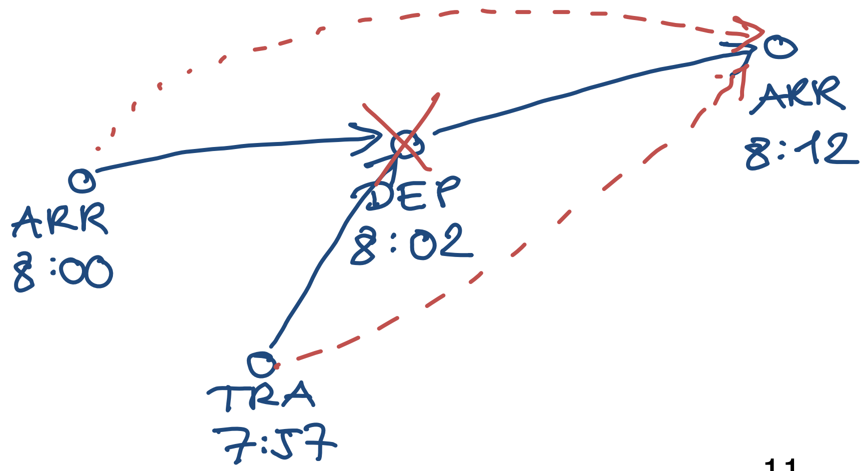
- Step 2: After the parse, add the missing arcs
  - For each station: sort the nodes by time, and for equal times, sort the transfer nodes before the departure nodes, with ties between nodes of the same kind broken arbitrarily
  - Then for each **transfer** node x in the sorted sequence
    - add an arc to the next transfer node in the sequence
    - add an arc to each departure node that comes after x without another transfer node inbetween (none, if next node after x is a transfer node)



Arrival, departure, transfer nodes 4/4

- Optimizations
  - If a station has several arrival nodes at the same time, it suffices to add a single transfer node for all of them
  - We can trivially **contract** all departure nodes: this decreases the number of arcs that were incident to the departure nodes by a factor of 3/2





Road vs. Transit Networks

- Assume the time-expanded model
  - Then we can run all our algorithms so far also for transit networks
  - But will the speed-up over ordinary Dijkstra be the same?
  - More about this in the next lecture

Multi-criteria cost functions 1/5

- So far our costs were always scalar numbers
  - ... namely the travel time
  - But there are many other criteria a user might want to optimize, too:
    - price (both road and transit networks)
    - beauty of the trip (both road and transit networks)
    - minimize walking between stations (transit only)
    - minimize number of transfers (transit only)
  - For the sake of explanation let us look at **two criteria** costs for the rest of the lecture: travel time and penalty

    (the penalty grows with more walking and more transfers)

Multi-criteria cost functions 2/5

- More than one solution
  - With two (or more) criteria, there is now the possibility of more than one optimal solution

    3 hours with 0 transfers is incomparable to

    2 hours with 1 transfer
  - However, some solutions are strictly better than others:

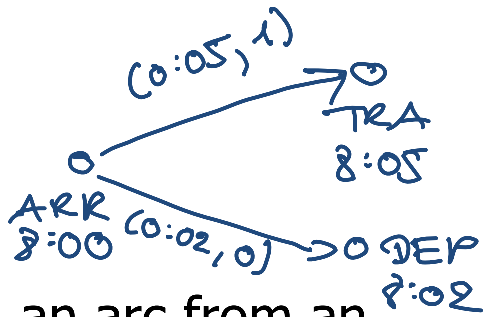
    2 hours with 1 transfer is better than

    3 hours with 2 transfers

Multi-criteria cost functions 3/5

- Formally
  - Costs are pairs (x, y) of scalars
  - We write $(x, y) \leq (x', y')$ if and only if $x \leq x'$ and $y \leq y'$
  - We write $(x, y) = (x', y')$ if and only if $x = x'$ and $y = y'$
  - We write $(x, y) < (x', y')$ iff $(x, y) \leq (x', y')$ and $(x, y) \neq (x', y')$
  - We write (x, y) (x', y') are incomparable

    if neither $(x, y) \leq (x', y')$ nor $(x', y') \leq (x, y)$



- Example
  - If the second component is simply #transfers, an arc from an arrival node at time 8:00 to a transfer node at time 8:05 would have cost (0:05, 1), and all other arcs would have costs (…, 0)

Multi-criteria cost functions 4/5

- Lemma
  - For each set of costs C there exists a subset C' of C such that
    - for each $c_1, c_2 \in C'$ with $c_1 \neq c_2$, c_1 is incomparable to c_2
    - for each $c \in C$, there exists a $c' \in C'$ with $c' \leq c$
  - Proof: as long as C contains c_1, c_2 with $c_1 \leq c_2$, remove c_2
- For a given query
  - ... let C be the set of costs of **all** possible paths
  - Then we want to compute a subset C' like above, called the **set of optimal solutions** or the **Pareto set** of C
  - As usual, we discuss only how to obtain the costs, and it will be easy to see in the end how to get paths with these costs

Multi-criteria cost functions 5/5

- For a given C, is this subset C' unique?
  - Let C_1 and C_2 be two subsets of optimal solutions

$$c_1 \in C_1 \Rightarrow c_1 \in C \overset{C_2 \text{ Pareto}}{\Longrightarrow} \exists c_2 \in C_2 \quad c_2 \leq c_1$$

$$c_2 \in C \overset{C_1 \text{ Pareto}}{\Longrightarrow} \exists c_1' \in C_1 \quad c_1' \leq c_2$$

$$c_1' \leq c_2 \leq c_1 \Rightarrow c_1' \leq c_1$$

$$C_1 \text{ Pareto} \Rightarrow c_1' = c_1 \Rightarrow c_2 = c_1 = c_1'$$

$$\Rightarrow c_1 \in C_2 \Rightarrow C_1 \subseteq C_2 \text{ and } C_2 \subseteq C_1$$

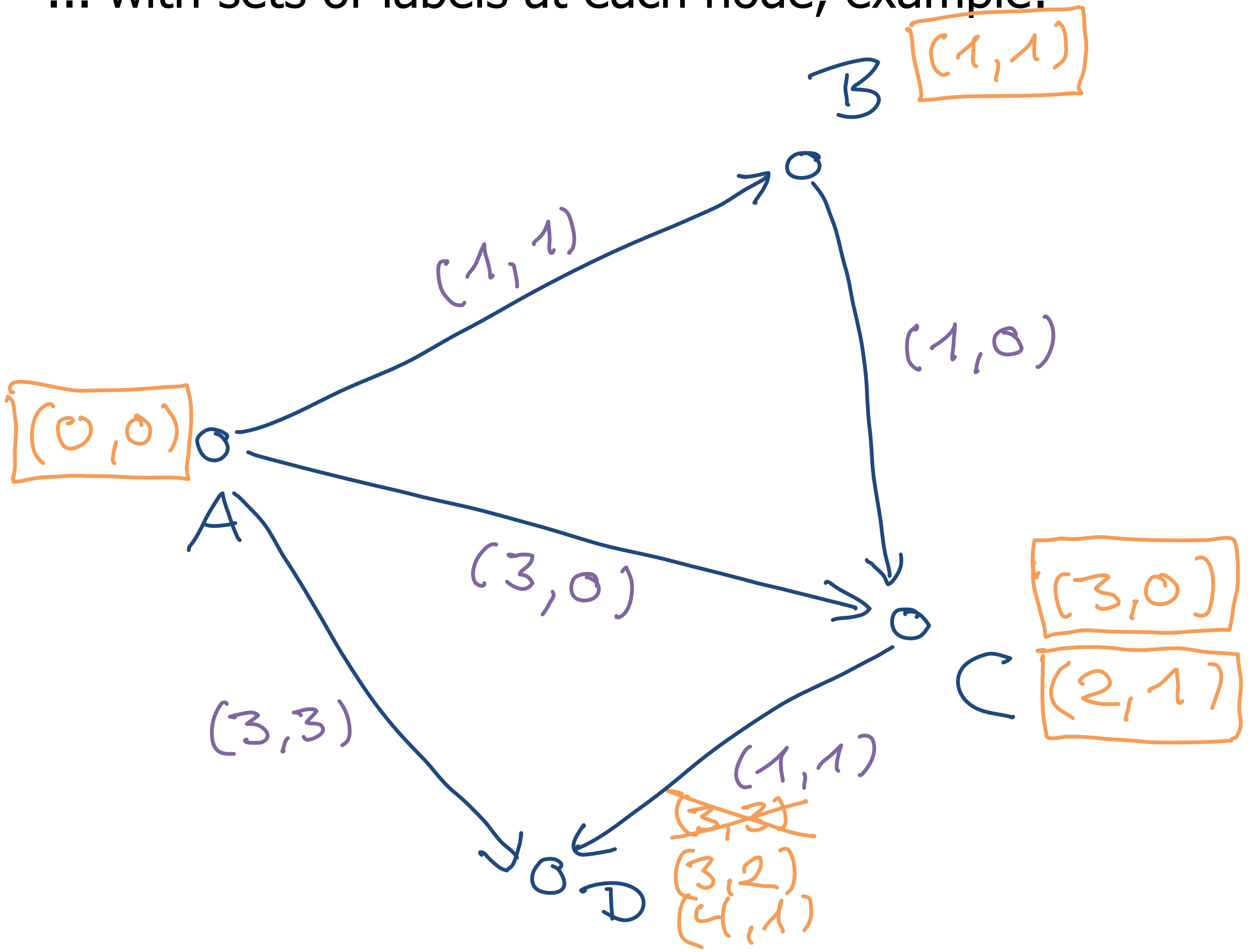
analogously

Multi-label Dijkstra 1/5

- How to compute these sets of solutions
  - Again, a variant of Dijkstra's algorithm does it
  - Consider ordinary Dijkstra, and think of the tentative costs at the nodes as **labels** (contain a single scalar, namely the tentative cost)
  - Initially there is only one label at the source, holding 0
  - All (not yet settled) labels are in a priority queue, according to some order on the set of possible labels
  - When processing the smallest label from the PQ, we settle it, and relax the outgoing arcs of the node to which it belongs, creating new labels at the adjacent nodes
  - At the adjacent nodes keep only the optimal labels

Multi-label Dijkstra 2/5

- We can do the exact same thing
  - ... with sets of labels at each node, example:



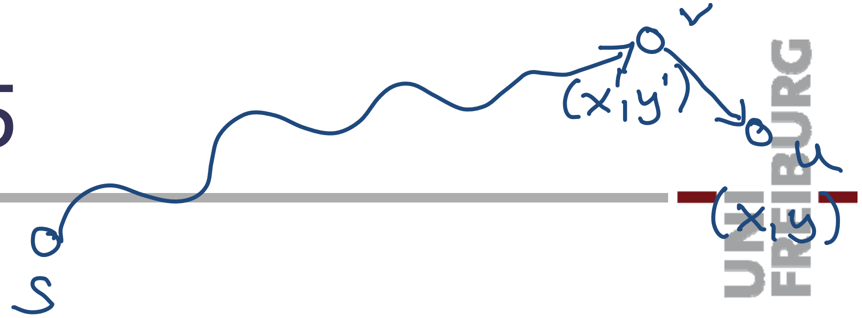
(5, 0) < (5, 1)

Multi-label Dijkstra 3/5

- In which order should we process the labels?
  - The order must be a **refinement** of the partial order we have for comparing labels, that is

    (x, y) < (x', y') ⇨ (x, y) must be processed before (x', y')
  - Why does it work? Why is that required? See next slide
  - For example, we can just look at the **first component** and if that is equal for two labels, look at the second comp.
  - Or we could also just look at the **second component** and if that is equal for two labels, look at the first comp.
  - Or we process by the order of the **sum of the components**

Multi-label Dijkstra 4/5

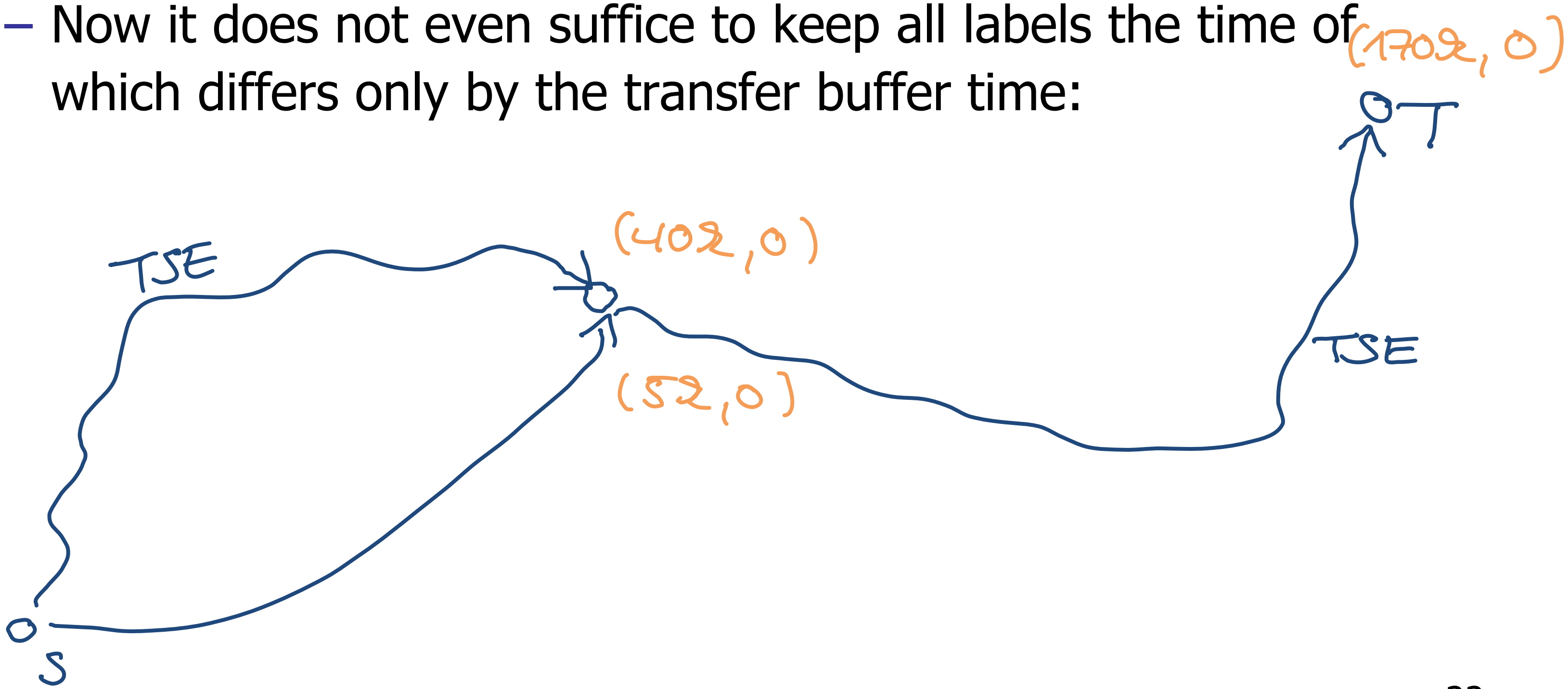


- Correctness proof (sketch)
  - For a given source node s, consider the union C of the sets of optimal costs from s at all nodes
  - As in our correctness proof for ordinary Dijkstra (Lecture 3), assume we have a strict order between all costs in C

    $(x_1, y_1) < (x_2, y_2) < (x_3, y_3) < \ldots$
  - Consider an arbitrary cost (x, y) from C at a node u, and let v be the predecessor of u of a shortest path to u with that cost
  - Let (x', y') be the cost of the path until v; note (x', y') < (x, y)
  - If the PQ order is a refinement of the label order, then (x', y') was processed earlier, and by way of induction everything was correct up to this point

Multi-label Dijkstra 5/5

- How about on a time-dependent graph?
  - Then we have a similar problem as with the transfer buffers
  - That is, labels computed along prefixes of shortest paths do not necessarily belong to shortest paths
  - Now it does not even suffice to keep all labels the time of which differs only by the transfer buffer time:



References

- Road Networks vs. Transit Networks

  Car or Public Transport — Two Worlds

  Hannah Bast, Efficient Algorithms 2009, LNCS 5760

  http://www.springerlink.com/content/y46257m66372x730/

- Multi-label Dijkstra

  Optimal paths in graphs with [...] multidimensional weights

  Ronald Prescott Loui, CACM 26(9), 1983

  http://portal.acm.org/citation.cfm?doid=358172.358406