On a blustery autumn evening five old friends met in the backroom of one of the city's oldest and most private clubs. Each had traveled a long distance — from all corners of the world — to meet on this very specific day… October 2, 1900 — 28 years to the day that the London eccentric, Phileas Fogg accepted and then won a £20,000 bet that he could travel Around the World in 80 Days.

When the story of Fogg's triumphant journey filled all the newspapers of the day, the five attended University together. Inspired by his impetuous gamble, and a few pints from the local pub, the group commemorated his circumnavigation with a more modest excursion and wager – a bottle of good claret to the first to make it to Le Procope in Paris.

Each succeeding year, they met to celebrate the anniversary and pay tribute to Fogg. And each year a new expedition (always more difficult) with a new wager (always more expensive) was proposed. Now at the dawn of the century it was time for a new impossible journey. The stakes: $1 Million in a winner-takes-all competition. The objective: to see which of them could travel by rail to the most cities in North America — in just 7 days. The journey would begin immediately…

Ticket to Ride is a cross-country train adventure. Players compete to connect different cities by laying claim to railway routes on a map of North America.

For 2 - 5 players
ages 8 and above
30 - 60 minutes

Patrick Schäfer
patrick.schaefer@hu-berlin.de

Semesterprojekt: Implementierung eines Brettspiels, WS 18/19
Agenda

• Today, 13:15
  – Competition
  – Short talk on Steiner Tree Approximation
Questions

• Are there any questions related to...
  – TTR-Server
  – TTR-Protocol
  – C#-Client-Implementation

• Benchmark AI and Shell-Script (also linked on the webpage):
  – https://box.hu-berlin.de/f/5b76e7c0f9084980ac63/?dl=1
Ticket to Ride and Graph Theory

- $G = (V,E)$, $V$=Cities, $E$ = Railways.
- Each vertex of the graph represents one city in Europe.
- An edge connects two cities.
- Each edge has a color and a length (cost).
- The graph contains more edges than any player can claim.
A MST for Ticket to Ride

The total weight of this minimum spanning tree is: 108
Minimum Spanning Tree (Forest)

• A spanning tree of the full graph would guarantee that any destination ticket is fulfilled.
• But payers do not have enough train tokens to claim a spanning tree of the full graph (45 vs 108).
• Thus, the best strategy is to capture a minimum spanning tree or forest of a subset of vertices (based on the destination tickets).

• **Steiner Tree / Forest:** Given an undirected, weighted graph $G=(V,E)$ and a subset of vertices $V'$, referred to as terminals, we search the subgraph $G'$ with minimum weight, that connects all terminals (and may include additional vertices).
Special Case: 2 terminals

1 destination ticket,
Simple: shortest path
Minimum-Weight Subtree

3 destination tickets, find the subgraph with minimum weight.
Minimum-Weight Subtree on Destination Tickets

3 destination tickets, find the subgraph with minimum weight.
Dijkstra

Total : $10 + 10 + 9 = 29$
Minimum-Weight Subtree on Destination Tickets

One cost-optimal Steiner tree

Total: 3+3+4+4+4+2+2=22
Approximate Algorithm

- Steiner Tree optimization problem is NP-hard, thus there is likely to be no exact polynomial time algorithm.
- There are *heuristic* algorithms with polynomial time, that have upper bound guarantees on the maximum cost.
- Implementing a good algorithm helps greatly for the AI-Challenge.
Steiner Tree Special Cases

- $|V'|=2$: Shortest Path
- $|V'|=V$: Compute the minimum spanning tree (MST)

- Idea: if we had a fully connected graph, then we can get the optimal solution using MST
  - Add edges to $G'$ that represent the shortest path between all nodes
Approximation Algorithm I

- G=(V,E), \(cost\) are the edge costs, S is a set of terminals
- Construct graph G’=(S,E’), \(cost’\)
  - build a fully connected graph:
    - for any pair of vertices \(cost’=\)distance of the shortest path
  - compute \(T = \text{MST}(G’)\)

- The minimum spanning tree T contains edges that represent shortest paths

- Recover Steiner Tree from T
  - \(T^* = \) recover shortest paths in G that correspond to edges in T
    - if present, remove cycles from \(T^*\)
The graph $G$
Build a fully-connected graph $G'$
A minimum spanning tree on $G'$
2-approximate Steiner tree
Approximation algorithm

• A $\alpha$-approximation algorithm is a polynomial-time algorithm that returns a solution of cost at most $\alpha$ times the cost of an optimal solution
2-approximation

- 2 x optimal costs
  = cost a DFS traversal (in blue) on the optimal tree that visits every edge exactly twice
  ≥ cost of some spanning tree constructed on shortest paths (orange)
  ≥ cost of the MST on orange edges
Minimum-Weight Subtree on Destination Tickets
Build a fully-connected graph $G'$
A minimum spanning tree on $G'$

Cost = 21
2-approximate Steiner tree

Cost = 5+8+8=21
An optimal (?) Steiner tree

Cost = 5+9+4=18
An alternative Approximation Algorithm

- $G= (V, E)$, $S$ is a set of terminals

- Heuristic for Steiner tree
  
  start with subset $T$ consisting of one terminal
  
  while $T$ does not span all terminals:
    
    select a terminal $t$ not in $T$ that is closest to a vertex in $T$
    
    add to $T$ the shortest path that connects $t$ with $T$

- Improve on solution
  
  build a subgraph of $G= (V, E)$ induced by the vertices in found solution $T$
  
  compute a MST
  
  repeat until there are no non-terminal leaves
    
    delete non-terminals that are leaves of the MST
select a terminal $t$ not in $T$ that is closest to a vertex in $T$
Approximation Algorithm II

(a) select a terminal \( t \) not in \( T \) that is closest to a vertex in \( T \)

(b)
Approximation Algorithm II

1. Select a terminal $t$ not in $T$ that is closest to a vertex in $T$.

2. Add $t$ to $T$.

3. Add all edges connecting $t$ to vertices in $T$.

4. Remove all edges connecting vertices in $T$.

Figure (a), (b), and (c) show the steps of the algorithm.

Select a terminal $t$ not in $T$ that is closest to a vertex in $T$.
Approximation Algorithm II

(a) select a terminal $t$ not in $T$ that is closest to a vertex in $T$
2-approximate Steiner Tree

build a subgraph of $G=(V,E)$ induced by the vertices in found solution $T$

Cost = $5 + 9 + 4 = 18$