Ticket to Ride: Concepts in Graph Theory

Patrick Schäfer
patrick.schaefer@hu-berlin.de

Semesterprojekt: Implementierung eines Brettspiels, WS 18/19
Agenda

• Last week
  – Technical refinement for new user stories
  –

• Today (w/ POs), 13:30
  – Sprint #1 Review Meeting; bring a laptop & presentable prototype
  – Sprint #2 kickoff; present your Sprint Backlog
  – short talk on
    • Ticket to Ride: concepts in graph theory
    • Coding guidelines / conventions
The Board Represents a Graph
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 has more edges than any player can claim
- The set of cities and edges is a player’s edge-induced subgraph
- Connected components: Player’s subgraphs don’t have to be connected

- Paths and Cycles
  - A destination ticket is met, when there is a path between the two cities
  - Creating cycles do not increase coverage, and thus do not help meeting destination tickets (but may block other players)
Graph Algorithms

• Concepts known from: “Algorithmen und Datenstrukturen”
  – Graph representations: Adjacency List, Adjacency Matrix
  – Shortest paths: Dijkstra, Floyd Warshall
  – Graph traversal: BFS, DFS
  – Minimum Spanning Tree: Prim, Kruskal
  – Topological sorting of directed graphs.
Ticket to Ride Rules and their Concepts in Computer Science

• What is the best representation of the board?
  – *Adjacency matrix* or *adjacency list*?

• Given a destination ticket, what is the shortest path?
  – *Shortest path*: *Dijkstra*

• How to fulfill destination tickets with the least amount of trains?
  – *Minimum spanning tree on subgraph* (*Minimum Steiner tree*)

• Calculating the final score:
  – List of routes claimed by a player
    • *Lookup in graph data structure* (*adjacency matrix* or *adjacency list*)
  – List of destination tickets fulfilled by a player.
    • *Graph traversal*: *DFS / BFS*
  – 10 point bonus awarded to player with the longest route on the board.
    • *Longest path in a tree / graph*
P, NP, NP-hard, NP-complete

• Definition:
  – P is the set of decision problems that can be solved in polynomial time
  – NP is the set of decision problems where we can verify a solution in polynomial time
  – NP-hard: at least as hard as NP (using polynomial time reduction)
  – NP-complete: it is NP-hard and in NP

What the world might look like...
From the rulebook:

„Finally, give the 10 point bonus for the European Express to the player(s) who have the Longest Continuous Path on the board. When evaluating and comparing path lengths, only take into account continuous lines of plastic trains of the same color. A continuous path may include loops, and pass through the same city several times, but a given plastic train may never be used twice in the same continuous path.“
How long is the longest simple path?
Acyclic Graph / Undirected Tree

How long is the longest simple path?

Simple...
Acyclic Graph / Undirected Tree

How long is the longest simple path?
Acyclic Graph / Undirected Tree

How long is the longest simple path?

Simple...
Longest Simple Path

• There is an algorithm for finding the longest simple path in undirected trees using two Depth-First-Searches:
  – Start DFS from a random vertex \( v \) and find the farthest vertex \( v' \) away.
  – Now, start a DFS from \( v' \) to find the vertex \( v'' \) farthest away from it. This path is the longest path in the graph.

• Does this still work in a cyclic graph?
Cyclic, Undirected Graph

How long is the longest simple path?

Not simple at all...
Cyclic, Undirected Graph

How long is the longest simple path?

Not simple at all...
Reformulation

• Is there a path in the player’s edge-induced subgraph, that visits all edges? **NP-hard**
  – Check for Euler path of the full graph is in **P**
  – But if there is no Eulerian path, we have to check $O(2^n)$ many subsets

• Finding the longest simple path in a cyclic graph is NP-hard. Thus, there is likely to be no polynomial time algorithm.

• There are approximate algorithms in polynomial time.
• For final scoring, we need the **exact length** of the longest path (not an approximation).

• Side note: finding the longest simple path in an undirected tree (acyclic graph) is in polynomial time.
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).
1 destination ticket, find the subgraph with minimum weight.

Easy: Dijkstra
Minimum-Weight Subtree on Destination Tickets

3 destination tickets, 6 cities, find the subgraph with minimum weight.

Dijkstra? MST? ...

MST: we don’t know which subset of nodes to include!

Dijkstra: single source shortest path for two cities.
Minimum-Weight Subtree on Destination Tickets

But do we really want to have a spanning tree (connect all routes)?
Minimum-Weight Subtree on Destination Tickets

Connected components
Minimum-Weight Subtree on Destination Tickets

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

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

Total: $10 + 10 + 9 = 29$

3 + 3 + 4

3 + 1 + 6

2 + 3 + 2 + 2
A Spanning Tree on the Subgraph...

Total: 3 + 2 + 2 + 3 + 1 + 6 + 4 + 4 = 25

Is it minimal?
A Spanning Tree on the Subgraph...

Is it minimal?

3 + 2 + 2 + 2 + 3 + 2 + 2 + 4 + 4 = 24
A Spanning Tree on the Subgraph…

Is it minimal?

3+3+4+4+4+2+2=22
NP-hardness

We are dealing with NP-hard optimization problems [1]:

- "LONGEST PATH": Given a non-negatively weighted graph \( G \) and two vertices \( u \) and \( v \), what is the longest simple path from \( u \) to \( v \) in the graph? A path is simple if it visits each vertex at most once.

- "STEINER TREE": Given a weighted, undirected graph \( G \) with some of the vertices marked, what is the minimum-weight subtree of \( G \) that contains every marked vertex? If every vertex is marked, the minimum Steiner tree is just the minimum spanning tree; if exactly two vertices are marked, the minimum Steiner tree is just the shortest path between them.

Steiner Tree / Forest in Cyclic Graphs is NP-hard

• Steiner Tree optimization problem is NP-hard, thus there is likely to be no exact polynomial time algorithm.

• Naïve approach:
  
  for each subset of nodes: \(2^{|V|}\)-Subsets
  
  compute the MST. \(O(|E|+|V|\log|V|)\)

  pick the subset with minimum costs.

• There are _heuristic_ algorithms with polynomial time, that have upper bound guarantees on the maximum cost.

• Finding a good algorithm is part of the AI-Challenge.
Literature


• Taking Students Out for a Ride: Using a Board Game to Teach Graph Theory: [p367-lim.pdf](http://www.cs.xu.edu/csci390/13s/p367-lim.pdf)