Assignment 1: Find the best train connection

AI-1 Systems Project (Winter Semester 2024/2025)

Jan Frederik Schaefer

Friedrich-Alexander-Universität Erlangen-Nürnberg, Department Informatik

Topic:	Search
Due on:	December 8, 2024
Version from:	October 11, 2024
Author:	Jan Frederik Schaefer

1 Task summary

Using a data set of Indian railway times, find the best connection between any two places according to different cost functions.

Didactic objectives

1. Gain some experience working with graphs,
2. solve a search problem for a large, real dataset,
3. learn how to implement an algorithm with efficiency in mind,
4. get some experience with how different cost functions affect a search algorithm,
5. get to know the CSV file format.

Prerequisites and useful methods

1. Search algorithms in general (as discussed in the AI lecture),
2. Dijkstra's algorithm [DA].

2 The data set

The train schedule is specified in the file schedule.csv in the assignment repository [AR]. There is also a mini-schedule.csv, which contains a smaller, more managable subset of schedule.csv. The schedule data is a modified variant of a data set from Kaggle [IRT]. It is stored as a CSV file with 12 columns, but only the following columns are relevant for us:

- **Train No.**: an identifier for the train.

- **islno**: what stop of the train is described (e.g. the fifth stop).
- **station Code**: an identifier for the train station.
- **Arrival time**: the arrival time at that stop.
- **Departure time**: the departure time at that stop.
- **Distance**: the total distance travelled until that stop (i.e. since the stop where islno is 1).

For example, let us take a look at the following two entries:

Train No.	islno	station Code	Arrival time	Departure time	Distance
04407	10	GD	23:30:00	23:35:00	536
04407	11	LKO	02:25:00	02:35:00	653

From this we learn that the 10th stop of train 04407 is Gonda Jn (GD) and that the next (11th) stop is Lucknow Nr (LKO). We can also see that the train travels $653 - 536 = 117$ kilometers from GD to LKO, which takes 2 hours and 50 minutes. Note that the arrival time 02:25:00 must refer to the next day. In general we will always need to "add a day" if the arrival time at stop n is smaller than the departure time at stop $n - 1$. We will do the same for the departure time at any stop if it is smaller than the arrival time, which is relevant for some of the cost functions.

3 Problems and solutions

Aside from the schedule data, you have a file problems.csv that contains the connection problems you have to solve. We also provide example problems (example-problems.csv) and solutions (example-solutions.csv), which you can use for comparison. The assignment repository [AR] has a script for checking your solutions for the example problems. A problem file has the following columns:

1. **ProblemNo**: the number of the problem.
2. **FromStation**: where the connection should start.
3. **ToStation**: where the connection should end.
4. **Schedule**: the schedule file (schedule.csv or mini-schedule.csv).
5. **CostFunction**: the cost function (Section 3.2).

You may assume that FromStation is different from ToStation and that ToStation is indeed reachable from FromStation. A solution file has three columns:

1. **ProblemNo**: the number of the problem solved.

Train No.	islno	station Code	Arrival time	Departure time	Distance
56502	57	SYM	14:34:00	14:35:00	588
56502	58	VLE	14:44:00	14:45:00	596
...					
57305	69	VLE	05:16:00	05:17:00	559
57305	70	SAB	05:22:00	05:23:00	562
57305	71	MUK	05:32:00	05:33:00	570
57305	72	NRT	06:00:00	06:02:00	578

Table 1: Relevant schedule data for connection 56502 : 57 -> 58 ; 57305 : 69 -> 72.

2. **Connection**: an optimal connection (usually not unique). The format is described in Section 3.1.
3. **Cost**: the cost of the solution according to the cost function.

3.1 Connection Format

The train connections have to be specified in a particular format. As an example, we will take a look at the following connection:

56502 : 57 -> 58 ; 57305 : 69 -> 72

x : y -> z means that we take train x from stop y (islno) until stop z. Semicolons separate trains taken consecutively. So, in the example, we would first take train 56502 from stop 57 to stop 58 and then continue with train 57305 on stop 69 until stop 72. This obviously requires that stop 58 of train 56502 is the same station as stop 69 of train 57305. Comparing with Table 1, we can see that it is a valid connection from SYM to NRT, with a change at VLE.

3.2 Cost functions

This section discusses the different cost functions, re-using the example connection

56502 : 57 -> 58 ; 57305 : 69 -> 72

and the schedule data from Table 1.

stops The number of times we enter a station by train. In the example, we would enter the stations VLE, SAB, MUK, NRT (i.e. we don't count the station we started from). The cost is thus 4.

timeintrain The total amount of time spent in a *moving* train in seconds (for simplicity, we ignore the time a train is in a train station and the time when trains are changed). In the example connection, we spend 9 minutes travelling in the first train and $5+9+27=41$ minutes travelling in the second train. The cost of the connection is therefore $(9+41)\cdot 60 = 3000$.

arrivaltime HH:MM:SS The time of arrival (including days) if you start at HH:MM:SS. So HH:MM:SS is basically the depature time and and the cost is the time of arrival (which we want to minize). Though a bit confusing, it is actually one of the most relevant cost functions (e.g. "it's 12:15 and I want to get to ...as soon as possible"). For example, with arrivaltime 11:30:00, we would arrive at 06:00:00 on the next day when using the example connection. The arrival time should thus be specified as 01:06:00:00. But with arrivaltime 15:24:00, we would miss the train and the arrival would be one day later (02:06:00:00).

traveltime The total time spent travelling in seconds, including the time spent in train stations. In the example connection, we would leave at "14:35:00" and arrive at "06:00:00" the next day. We would therefore spend 15 hours and 25 minutes travelling, which is 55500 seconds.

4 What to submit

Your solution should be pushed to your gitlab repository for this assignment. Concretely, the repository should contain:

1. all your code for solving this assignment,
2. a README.md file explaining
   i. dependencies (programming language, version, external libraries and how to get them),
   ii. how to run your code to solve other problems,
   iii. the repository structure,
   iv. anything else we should know,
3. a solution summary (see [SoS] for more details – it should describe the main ideas, not document the code),

4. a file solutions.csv that contains your solutions for the problem file (problems.csv) as specified in Section 3.

5 Points

The total number of points for this assignment is 100. You can get up to 20 points for the quality of the submission (README, evaluation, ...). Furthermore, you will get 1 point for every correct entry in solutions.csv, which means that you can get up to 80 points for the solutions. For non-optimal entries in the solutions.csv that are otherwise correct (valid connection and correct cost) you will get $\frac{1}{2}$ point.

If the grading scheme doesn't seem to work well, we might adjust it later on (likely in your favor).

References

[AR] *Repository for Assignment 1: Find the best train connection.* URL: https://gitlab.rrze.fau.de/wrv/AISysProj/ws2425/a1.1-find-train-connections/assignment.

[DA] *Dijkstra's algorithm.* URL: https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm (visited on 10/27/2022).

[IRT] *Indian Railways Time Table for trains available.* URL: https://www.kaggle.com/harsh16/indian-railways-time-table-for-trains-available (visited on 11/01/2021).

[SoS] *Solution Summary.* URL: https://gitlab.rrze.fau.de/wrv/AISysProj/admin/general/-/blob/main/solution-summary.md.