



**UNIVERSITY OF MACEDONIA**  
MSc in Artificial Intelligence and Data  
Analytics

# Train-scheduling for railway networks using basic kinematics and A\*

Dimitrios Manolakis

Supervisor: Ioannis Refanidis

# Introduction

- ▶ Goal: Automation of train-scheduling for railway networks.
- ▶ First algorithm: Computing optimal speed profile over a given path
  - Makes use of various train and path characteristics.
  - Optimizes journey duration.
  - Basic kinematic equations are applied.
  - Approximates real-world conditions.
- ▶ Second algorithm: Time-dependent shortest path finding
  - A\* search
  - Straight line distance admissible heuristic function

# Related Work

- ▶ Train scheduling is considered an NP-complete problem.
- ▶ Operations research techniques:
  - branch and bound algorithm
  - Lagrangian relaxation decomposition
  - linear programming
  - tabu search
  - Dijkstra algorithm
  - genetic algorithms
  - ant colony optimization
- ▶ Machine learning techniques:
  - Reinforcement learning (Q-learning algorithm)
  - Deep neural networks

# Problem Formulation (1/2)

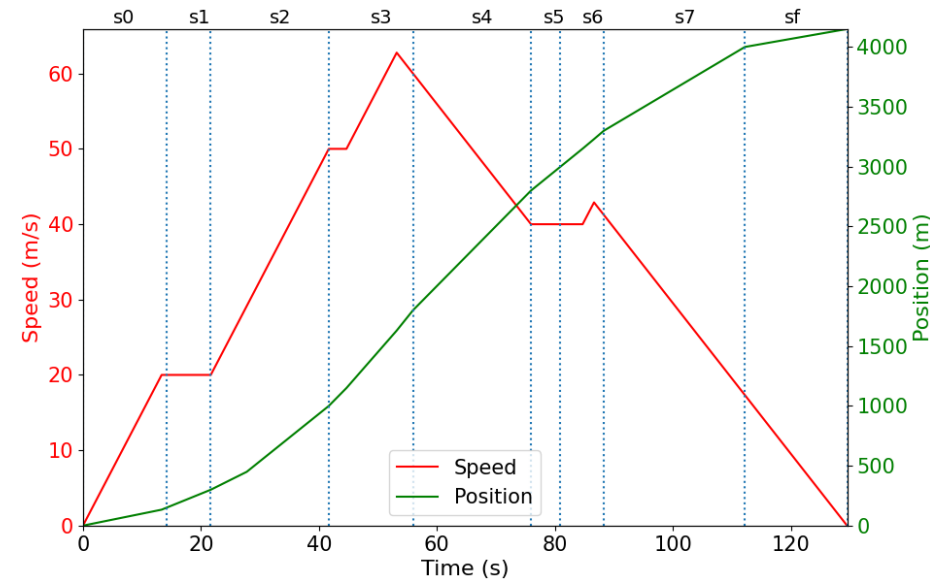
- ▶ Railway network graph modeled as an undirected graph  $G = (V, E)$ .
- ▶ Each path is composed of multiple segments.
- ▶ Path segment characteristics.
  - length
  - maximum allowed speed
- ▶ Train characteristics
  - length
  - maximum speed
  - acceleration
  - deceleration
- ▶ State
  - position
    - list of segments
    - remaining distance
  - speed

# Problem Formulation (2/2)

- ▶ Problem definition: initial and goal state
- ▶ Optimal solution: minimum duration speed profile

Parameters	Value
Length $l(T)$	150 m
Maximum speed $v_{max}(T)$	75 m/s
Acceleration $max_{acc}(T)$	1.5 m/s <sup>2</sup>
Deceleration $max_{dec}(T)$	1 m/s <sup>2</sup>
<i>init</i>	$((s_0, 150), 0)$
<i>goal</i>	$((s_f, 0), 0)$

Segment $s$	Length $l_s$ (m)	$v_{max}(s)$ (m/s)
$s_0$	150	20
$s_1$	150	40
$s_2$	700	50
$s_3$	800	80
$s_4$	1000	85
$s_5$	200	40
$s_6$	300	60
$s_7$	700	50
$s_f$	150	30



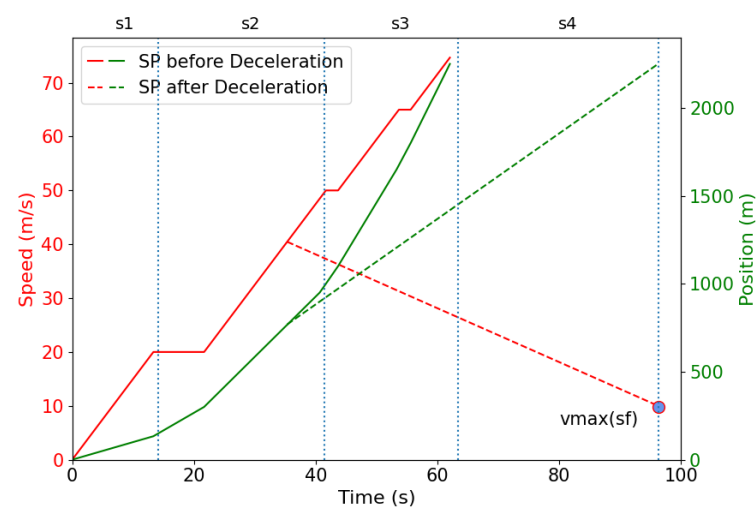
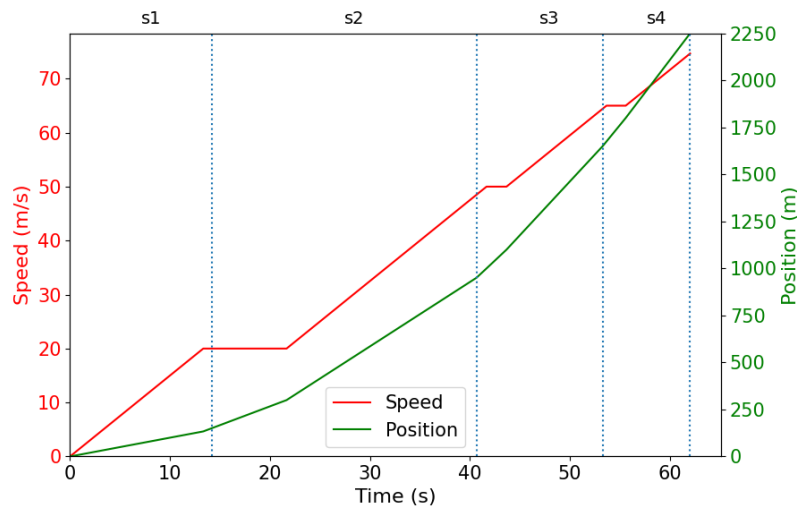
# Computing optimal speed profile over a given path (1/6)

- ▶ Input: Path & Train characteristics, initial and goal state.
- ▶ Output: Optimal speed profile.
- ▶ An event-based approach is applied.
- ▶ An event happens when:
  - the train's head enters a new segment.
  - the train's tail exits a segment.
  - the train reaches the (segment or train) speed limit.
- ▶ The time to reach each type of event is calculated and the speed profile is adapted accordingly.
- ▶ **Difficult case:** When entering a new segment, there might a be speed limit lower than the train's current speed.

# Computing optimal speed profile over a given path (2/6)

Segment $s$	Length $l_s$ (m)	$v_{max}(s)$ (m/s)
$s_0$	150	20
$s_1$	150	20
$s_2$	700	65
$s_3$	800	70
$s_4$	1000	78
$s_f$	150	10

Parameters	Value
Length $l(T)$	150 m
Maximum speed $v_{max}(T)$	78 m/s
Acceleration $max_{acc}(T)$	1.5 m/s <sup>2</sup>
Deceleration $max_{acc}(T)$	-0.5 m/s <sup>2</sup>
$init$	$((s_0, 0), 0)$
$goal$	$((s_f, 0), 0)$



# Computing optimal speed profile over a given path (3/6)

Init & goal states

```
Paper_Example_2 Notepad
File Edit Format View Help
init = (([1],0),0)
goal = (([6],0),0)
trainLength = 150
maxAcc = 1.5
maxDec = 0.5
trainMaxSpeed = 78
maxSpeeds = [20,20,50,65,85,10]
positions = [150,300,1100,1800,2400,2550]
```

Train characteristics

Path characteristics



# Computing optimal speed profile over a given path (4/6)

Input data validation

```
D:\Dissertation>py OptimalPathSpeedProfile.py D:\Dissertation\Paths\Paper_Example_2.txt  
Train characteristics are valid.  
Path characteristics are valid.  
Init and Goal states are valid.
```

```
Total time: 121.2427603749243
```

Path travel time

```
Optimal Speed profile of the path
```

```
State 1: (([1, 2], 150), 0), Time: 0  
State 2: (([1, 2], 16.666666666666667), 20.0), Time: 13.333333333333334  
State 3: (([2, 3], 800), 20.0), Time: 14.166666666666666  
State 4: (([3], 650.0), 20.0), Time: 21.666666666666664  
State 5: (([3], 237.5), 40.46603514059662), Time: 35.310690093731075  
State 6: (([5, 6], 150), 10), Time: 96.2427603749243  
State 7: (([5, 6], 100.0), 10.0), Time: 101.2427603749243  
State 8: (([6], 0), 0), Time: 121.2427603749243
```

Optimal speed profile

# Computing optimal speed profile over a given path (5/6)

State of the train at the start of the Iteration

Time needed until each event

```
D:\Dissertation>py OptimalPathSpeedProfile.py D:\Dissertation\Paths\Paper\Example_2.txt --debug
Train characteristics are valid.
Path characteristics are valid.
Init and Goal states are valid.
-----
iterationCounter 1
speedLimit: 20, currentAcc: 1.5, t1, t2, t3: (14.142135623730951, 14.142135623730951, 13.333333333333334)
current State: (([1, 2], 150), 0)
current time: 13.333333333333334
hExit: 16.666666666666657, tExit: 16.666666666666657
Train moved 133.33333333333334 meters in 13.333333333333334 seconds, CurrentV: 20.0, currentAcc: 1.5
New state: (([1, 2], 16.666666666666657), 20.0)
-----
```

State of the train at the event

# Computing optimal speed profile over a given path (6/6)

```
D:\Dissertation>py OptimalPathSpeedProfile.py D:\Dissertation\Paths\Impossible_to_decelerate.txt
Train characteristics are valid.
Path characteristics are valid.
Init and Goal states are valid.

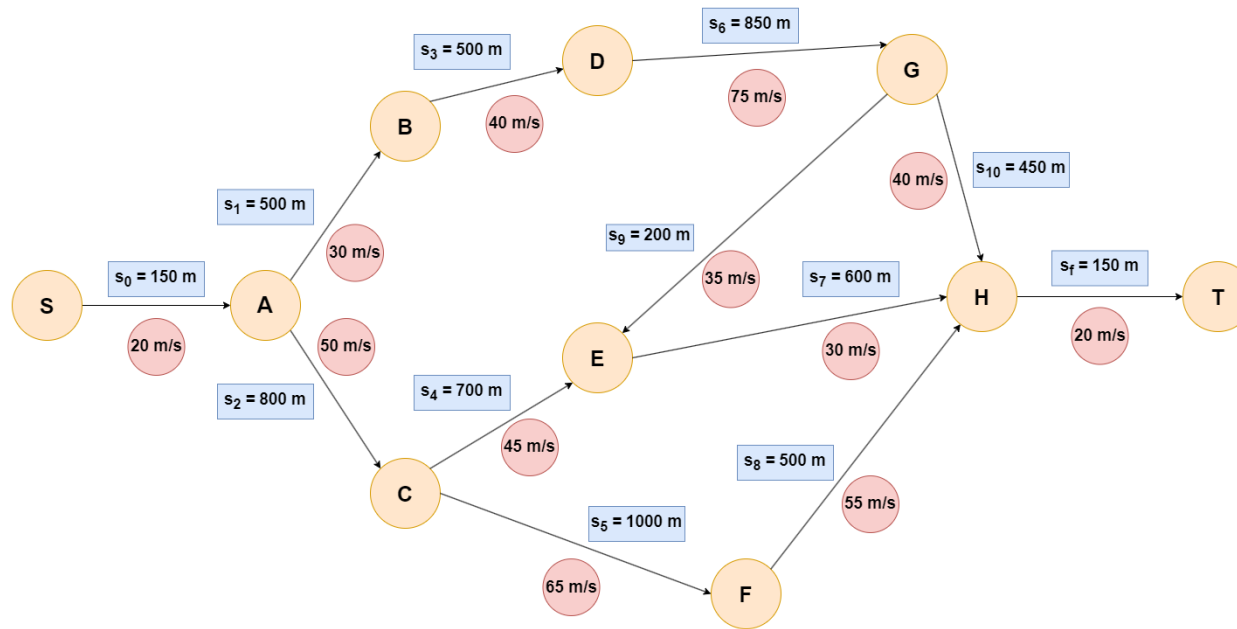
No deceleration point could be found. There is no solution!

D:\Dissertation>py OptimalPathSpeedProfile.py D:\Dissertation\Paths\Impossible_to_accelerate.txt
Train characteristics are valid.
Path characteristics are valid.
Init and Goal states are valid.

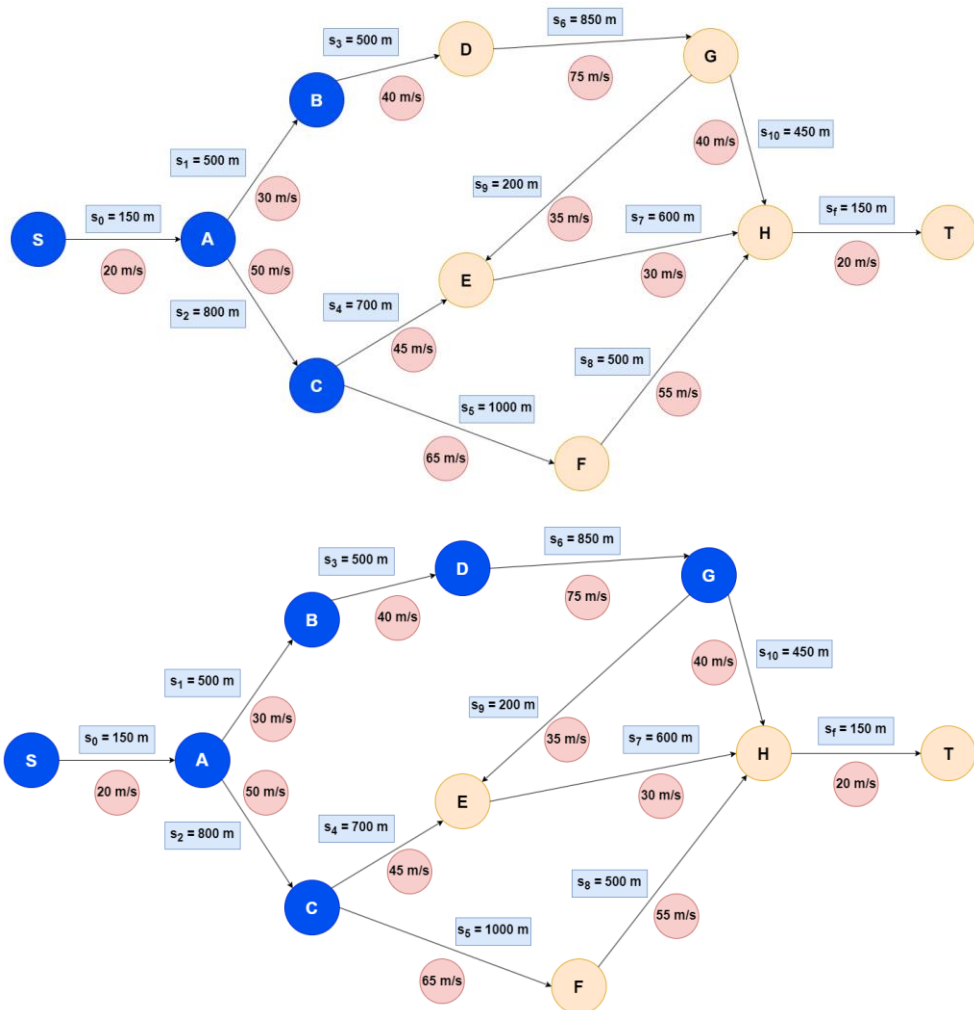
It is impossible to accelerate to goalV!
```

# Shortest path finding (1/7)

- ▶ Input: G, Train characteristics, initial and goal state.
- ▶ Output: Optimal speed profile for the shortest path.
- ▶ A\* search with a straight-line heuristic.



# Shortest path finding (2/7)



- ▶ Detected paths are expanded until the nearest junction where multiple paths meet.

- Paths [S,A,B] and [S,A,C] are detected.
- Path [S,A,B] is extended into path [S,A,B,D,G].

- ▶ Graph weights:  $f = g + h$

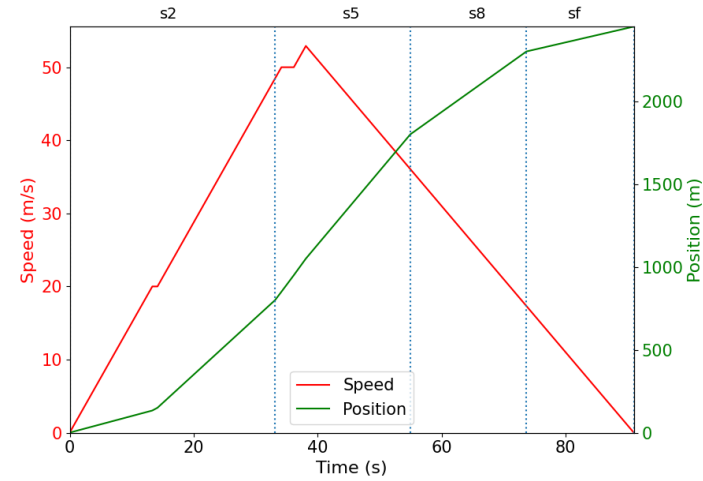
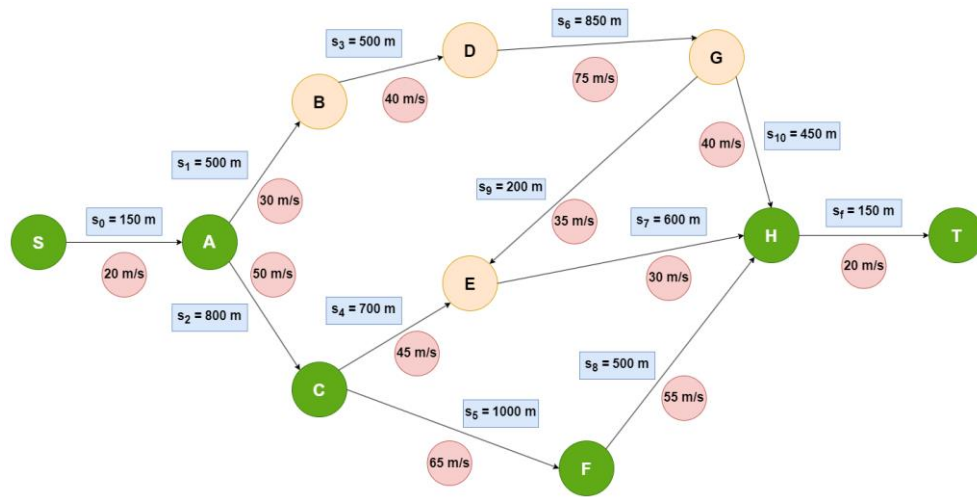
- G values:

$$G([S,A,B,D,G]) = 59.16 \text{ seconds}$$

$$G([S,A,C]) = 33.15 \text{ seconds}$$

- H values are the time needed to travel the straight line distance between the last node of a path and the destination node using the train's maximum speed.

# Shortest path finding (3/7)



# Shortest path finding (4/7)

```
Paper_Example_3_Characteristics - N...
File Edit Format View Help
init = (([1],0),0)
goal = (([12],0),0)
trainLength = 150
maxAcc = 1.5
maxDec = 1
trainMaxSpeed = 75
Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

```
Paper_Example_3_Network - Notepad
File Edit Format View Help
$ A {'label':0,'maxspeed': 20, 'length':150, 'straight_line': 2000}
A B {'label':1,'maxspeed': 30, 'length':500, 'straight_line': 1400}
A C {'label':2,'maxspeed': 50, 'length':800, 'straight_line': 1200}
B D {'label':3,'maxspeed': 40, 'length':500, 'straight_line': 900}
C E {'label':4,'maxspeed': 45, 'length':700, 'straight_line': 900}
C F {'label':5,'maxspeed': 65, 'length':1000, 'straight_line': 500}
D G {'label':6,'maxspeed': 75, 'length':850, 'straight_line': 500}
E H {'label':7,'maxspeed': 30, 'length':600, 'straight_line': 50}
F H {'label':8,'maxspeed': 55, 'length':500, 'straight_line': 50}
G E {'label':9,'maxspeed': 35, 'length':200, 'straight_line': 900}
G H {'label':10,'maxspeed': 40, 'length':450, 'straight_line': 50}
H T {'label':11,'maxspeed': 20, 'length':150, 'straight_line': 0}
Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

# Computing optimal speed profile over a given path (5/7)

Nodes of the shortest path

```
D:\Dissertation>py ShortestPath.py D:\Dissertation\Networks\Paper_Example_3_Network.txt D:\Dissertation\Networks\Paper_Example_3_Characteristics.txt
Shortest path is: ('S', 'A', 'C', 'F', 'H', 'T')
Total time: 91.02504370215301
Optimal Speed profile of the path
State 1: (([1, 2], 800), 0), Time: 0
State 2: (([1, 2], 666.6666666666666), 20.0), Time: 13.333333333333334
State 3: (([2], 650.0), 20.0), Time: 14.166666666666666
State 4: (([2, 3], 1000), 48.47679857416329), Time: 33.15119904944219
State 5: (([2, 3], 950.0), 50.0), Time: 34.166666666666664
State 6: (([3], 850.0), 50.0), Time: 36.166666666666664
State 7: (([3], 750.0), 52.91502622129181), Time: 38.1100174808612
State 8: (([5], 0), 0), Time: 91.02504370215301
```

Shortest path travel time

Optimal speed profile of the shortest path



# Computing optimal speed profile over a given path (6/7)

Expansion of the current path

```
Current fastest path: ('S', 'A', 'C')
Generated paths: dict_keys([('S', 'A', 'C', 'E'), ('S', 'A', 'C', 'F')])
Expanded paths: dict_keys([('S', 'A', 'C', 'E', 'H', 'T'), ('S', 'A', 'C', 'F', 'H', 'T')])
Calculating g value on path ('S', 'A', 'C', 'E', 'H', 'T')
-----
iterationCounter: 1
speedlimit: 50, currentAcc: 1.5, t1, t2, t3: (21.94335081419454, 18.984532382775527, 20.0)
current State: (([2], 650.0), 20.0)
current time: 33.15119904944219
hExit: 0.0, tExit: 150.0
Train moved 650.0 meters in 18.984532382775527 seconds, CurrentV: 48.47679857416329, currentAcc: 1.5
New state: (([2], 0.0), 48.47679857416329)
Train reached a segment with speed limit 45 with a speed equal to 48.47679857416329.
i: 2, v0: 45, v1: 20.0, v2: 48.47679857416329, hExit: 650.0
d: 800, d1: 150.0, xDec: 735.0, v1Hash: 46.42197755374064
The train can accelerate until xDec. New state: (([2], 65.0), 46.42197755374064). Acc time: 17.61465170249376, Dec time: 1.4219775537406392.
-----
```

G value computation

# Computing optimal speed profile over a given path (7/7)

The g values of all paths are equal to infinite.

```
openDict with g values: {'S', 'A', 'B', 'D', 'G'): inf, ('S', 'A', 'C', 'E', 'H', 'T'): inf, ('S', 'A', 'C', 'F', 'H', 'T'): inf}
Calculating f value on path ('S', 'A', 'B', 'D', 'G')
g value: inf
heuristic: 3.3333333333333335
f value: inf
Calculating f value on path ('S', 'A', 'C', 'E', 'H', 'T')
g value: inf
heuristic: 0.0
f value: inf
Calculating f value on path ('S', 'A', 'C', 'F', 'H', 'T')
g value: inf
heuristic: 0.0
f value: inf
Data after f values computation.
openDict: {'S', 'A', 'B', 'D', 'G'): inf, ('S', 'A', 'C', 'E', 'H', 'T'): inf, ('S', 'A', 'C', 'F', 'H', 'T'): inf}
SPDict: {'S', 'A', 'B', 'D', 'G'): [0], ('S', 'A', 'C', 'E', 'H', 'T'): [0], ('S', 'A', 'C', 'F', 'H', 'T'): [0]}
timeDict: {'S', 'A', 'B', 'D', 'G'): [inf], ('S', 'A', 'C', 'E', 'H', 'T'): [inf], ('S', 'A', 'C', 'F', 'H', 'T'): [inf]}
The problem cannot be solved!
```

The f values of all paths become equal to infinite.

Hence, the problem cannot be solved!

# Conclusions

- ▶ Realistic problem formulation.
- ▶ Handles the train when at multiple segments.
- ▶ Backtracks in previous states when a deceleration is needed.
- ▶ Many more train or path characteristics can be added.
- ▶ Different optimization criteria can be used.
- ▶ Different algorithms can be applied for the fastest route search.
- ▶ Problem can be expanded to handle multiple trains in the railway network.
- ▶ Real-world data can be used.

# Repository

The screenshot shows a GitHub repository page for 'Finding-Time-Optimal-Routes-for-Trains-Using-Basic-Kinematics-and-A' by user Dimimano. The repository is public and has 1 branch and 0 tags. The file list includes: Networks, Paths, Dissertation.pdf, OptimalPathSpeedProfile.py, README.md, and ShortestPath.py. The README is expanded, showing the title 'Finding-Time-Optimal-Routes-for-Trains-Using-Basic-Kinematics-and-A\*' and a description: 'The code implementation of my dissertation for the AI & Data Analytics Master's program @ University of Macedonia.' The description continues with a detailed paragraph about the problem of finding time optimal routes for trains over a railway network, and another paragraph about the solution using basic kinematics and A\*. A citation is provided at the bottom: 'Manolakis, D., Refanidis, I. (2024). Finding Time Optimal Routes for Trains Using Basic Kinematics and A. In: Nowaczyk, S., et al. Artificial Intelligence. ECAI 2023 International Workshops. ECAI 2023. Communications in Computer and Information Science, vol 1947. Springer, Cham. [https://doi.org/10.1007/978-3-031-50396-2\\_7](https://doi.org/10.1007/978-3-031-50396-2_7)'.

**Repository Details:**

- Repository: Finding-Time-Optimal-Routes-for-Trains-Using-Basic-Kinematics-and-A (Public)
- User: Dimimano
- Branches: 1 Branch
- Tags: 0 Tags
- Commits: 4 Commits

**File List:**

File Name	Action	Last Modified
Networks	Add files via upload	last month
Paths	Add files via upload	last month
Dissertation.pdf	Add files via upload	last month
OptimalPathSpeedProfile.py	Add files via upload	last month
README.md	Update README.md	last month
ShortestPath.py	Add files via upload	last month

**README Content:**

## Finding-Time-Optimal-Routes-for-Trains-Using-Basic-Kinematics-and-A\*

The code implementation of my dissertation for the AI & Data Analytics Master's program @ University of Macedonia.

This study tackles the problem of finding time optimal routes for trains over a railway network. The problem is defined as follows: A train has a known length. The position of the train is defined over parts of one or more consecutive track segments. There are a maximum speed, a maximum acceleration and a maximum deceleration capability for the train. Each track segment has a maximum allowed speed for any train being over it. A problem instance is defined by an initial and a goal state, which are two positions accompanied with desired speeds (being usually, but not necessarily, zero). In this paper we are interested in minimizing the total duration of reaching the goal state from the initial one; other metrics such as fuel consumption could be considered.

We solve this problem using basic kinematics and A\*. We present two algorithms: The first one computes analytically in continuous space the optimal speed profile of the train for a problem defined over a given path. The second algorithm extends the first one over arbitrary graphs. A\* empowered with a simple admissible heuristic is employed to find the optimal combination of speed profile and path.

Manolakis, D., Refanidis, I. (2024). Finding Time Optimal Routes for Trains Using Basic Kinematics and A. In: Nowaczyk, S., et al. Artificial Intelligence. ECAI 2023 International Workshops. ECAI 2023. Communications in Computer and Information Science, vol 1947. Springer, Cham. [https://doi.org/10.1007/978-3-031-50396-2\\_7](https://doi.org/10.1007/978-3-031-50396-2_7)

**Repository Features:**

- About: The code implementation of my dissertation for the AI & Data Analytics Master's program @ University of Macedonia
- Releases: No releases published. [Create a new release](#)
- Packages: No packages published. [Publish your first package](#)
- Languages: Python 100.0%
- Suggested workflows: Based on your tech stack. Includes Python package, Pylint, and Python application workflows.

<https://github.com/Dimimano/Finding-Time-Optimal-Routes-for-Trains-Using-Basic-Kinematics-and-A->

Thank you for your time!  
Questions?