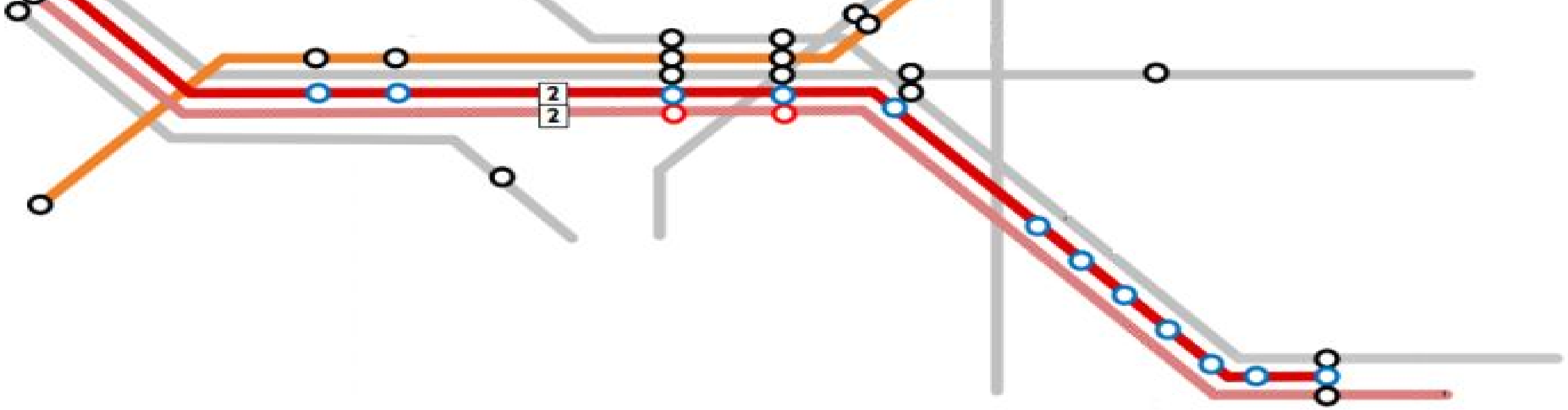


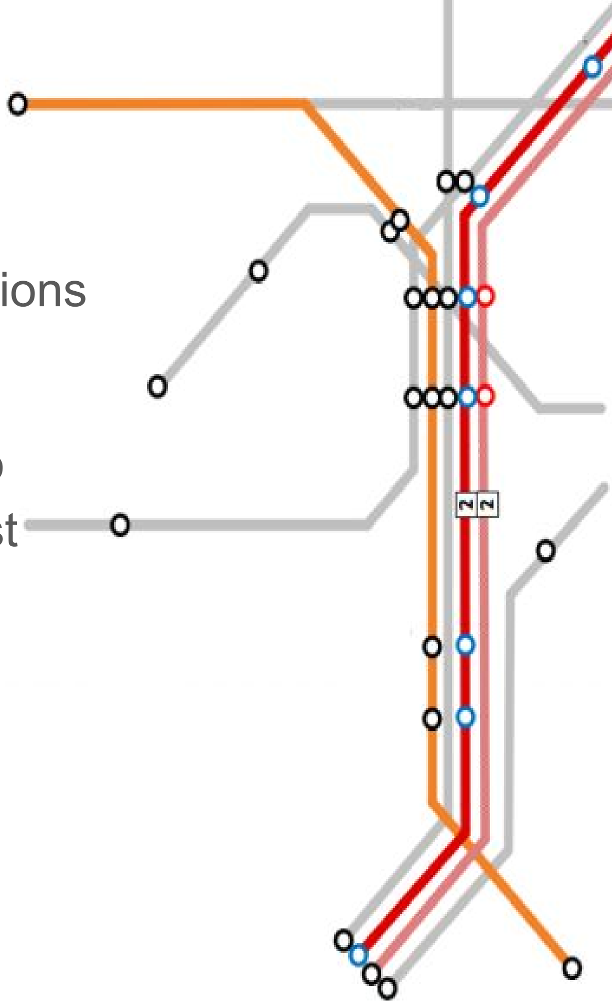
TRAIN DISPATCH

TIANGANG CHEN, ISABELLA DEMEO, RAJA SHAROSE KHAN

PROBLEM

The motivation behind this project can be found in the millions of dollars spent yearly on the train transportation network.

> Design a system that can route trains from location to location taking into account time optimization and cost efficiency.



PROBLEM

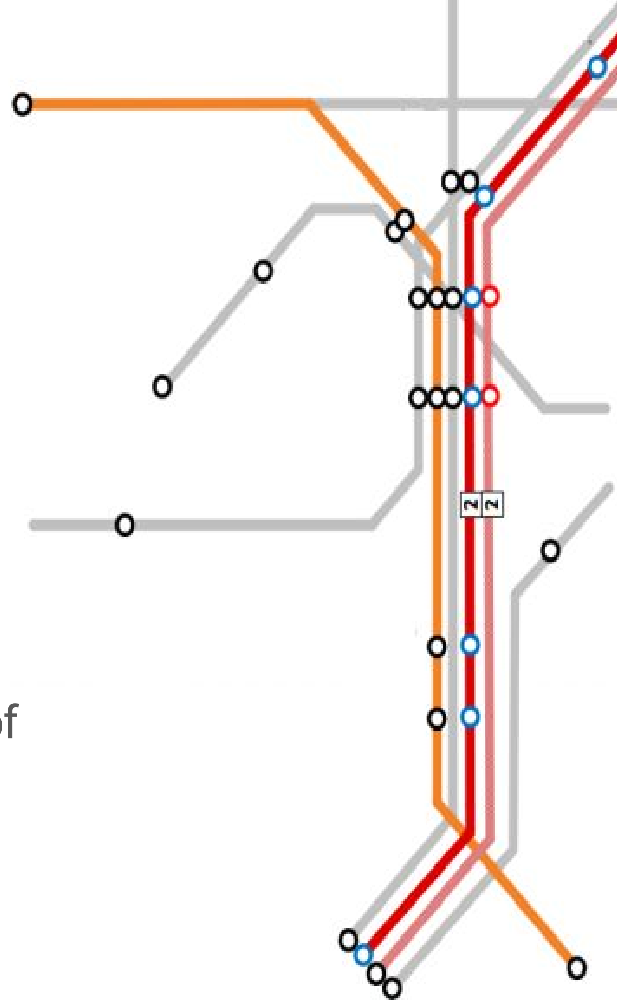
Routing trains in an undirected, weighted graph

Vertices/nodes: the train stations

Edges: the railways.

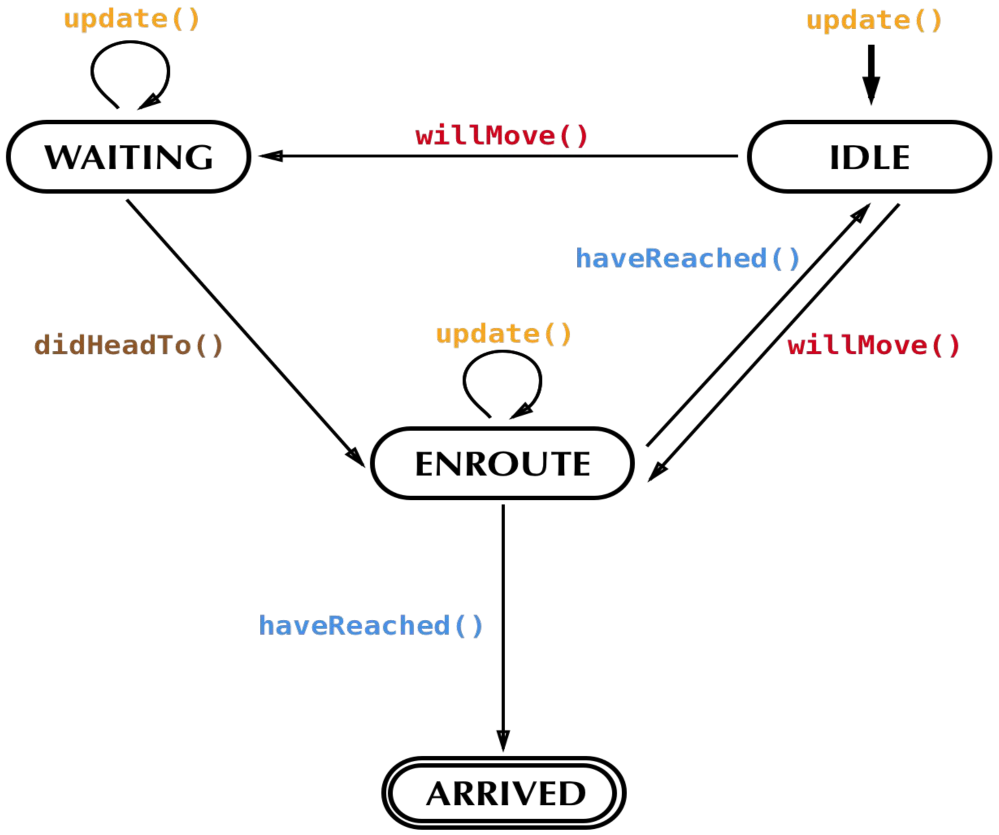
The railways for this train network are two-way tracks

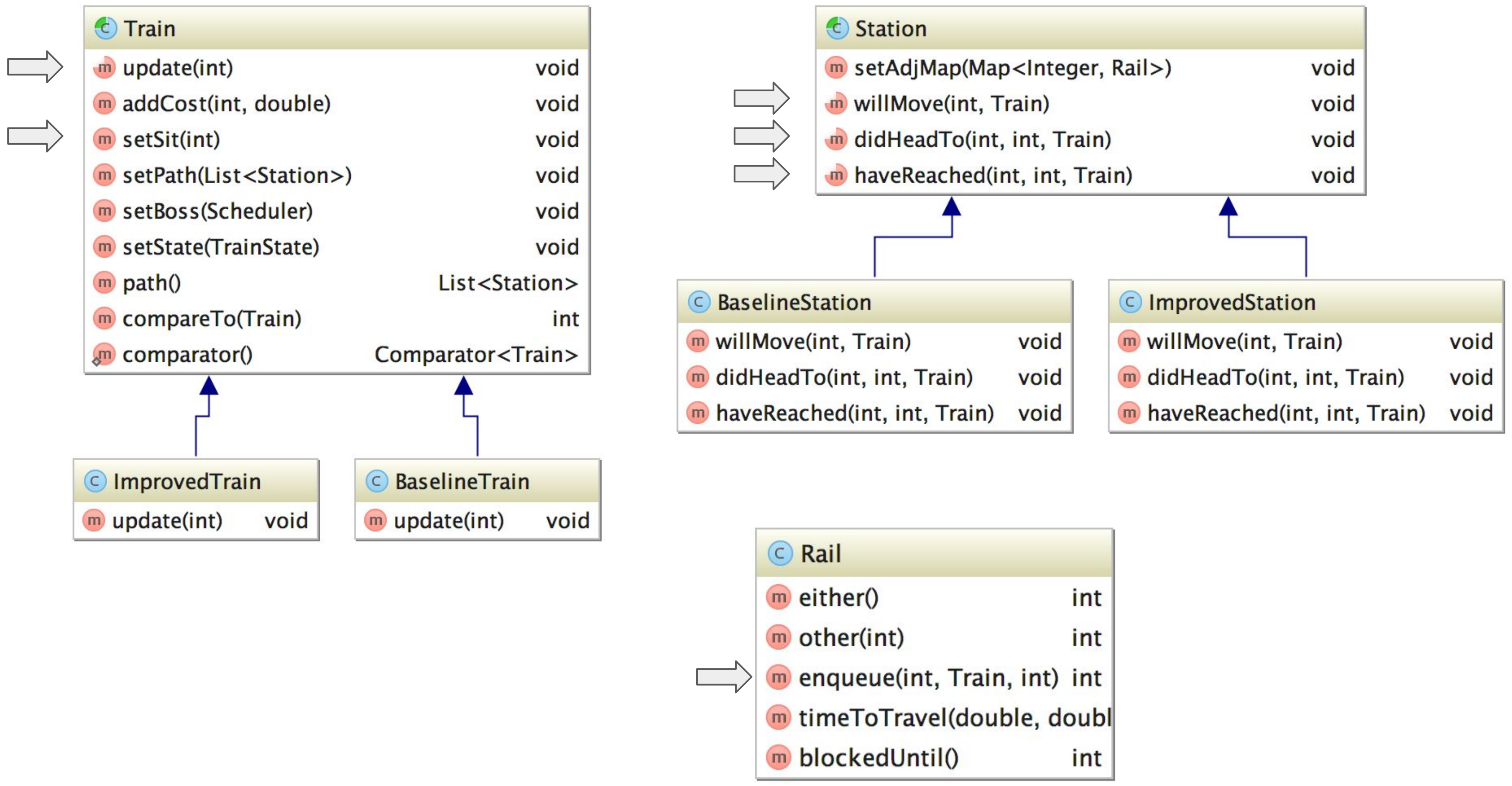
The trains running on this track have an infinite capacity of passengers



IMPLEMENTATION OF THE SOLUTION : overview

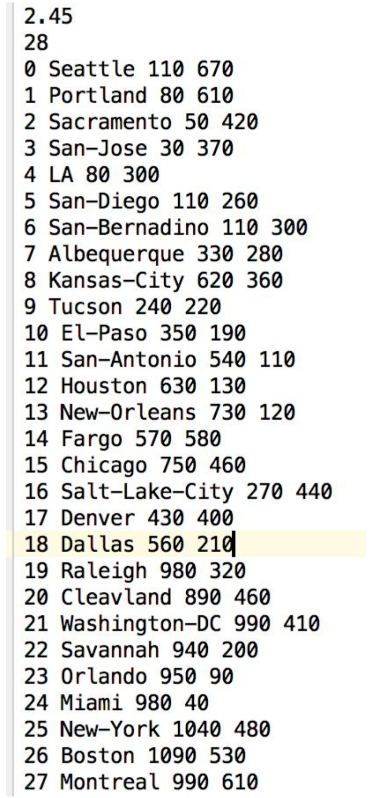
- DUMB Trains
- SMART Stations
- SUPER SMART Rails




Train
update(int) void
addCost(int, double) void
setSit(int) void
setPath(List<Station>) void
setBoss(Scheduler) void
setState(TrainState) void
path() List<Station>
compareTo(Train) int
comparator() Comparator<Train>
ImprovedTrain
update(int) void
BaselineTrain
update(int) void
Station
setAdjMap(Map<Integer, Rail>) void
willMove(int, Train) void
didHeadTo(int, int, Train) void
haveReached(int, int, Train) void
BaselineStation
willMove(int, Train) void
didHeadTo(int, int, Train) void
haveReached(int, int, Train) void
ImprovedStation
willMove(int, Train) void
didHeadTo(int, int, Train) void
haveReached(int, int, Train) void
Rail
either() int
other(int) int
enqueue(int, Train, int) int
timeToTravel(double, doubl
blockedUntil() int

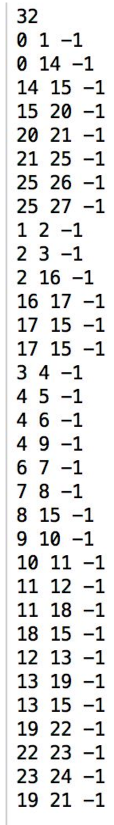
IMPLEMENTATION OF THE SOLUTION: INPUT

Stations

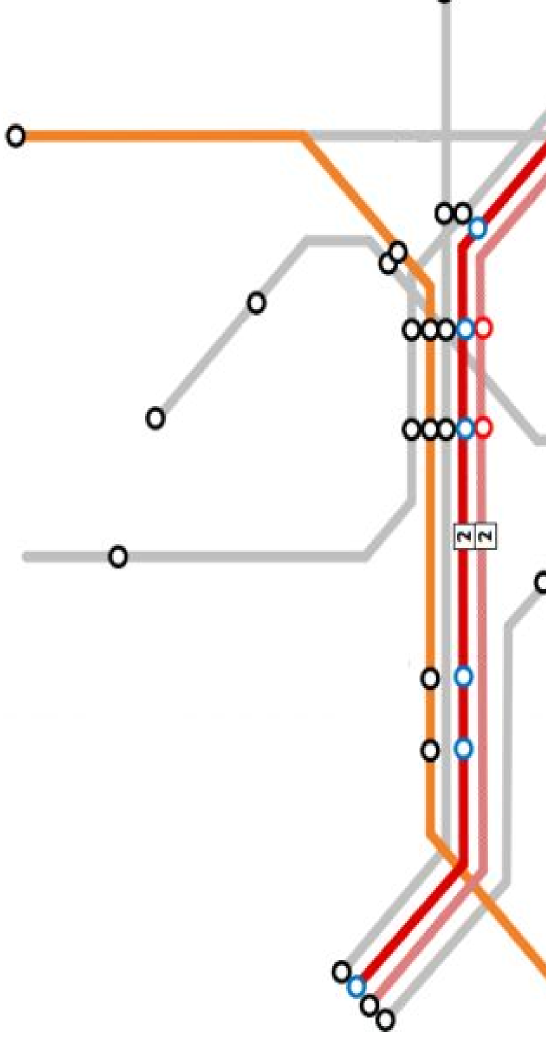


```
2.45
28
0 Seattle 110 670
1 Portland 80 610
2 Sacramento 50 420
3 San-Jose 30 370
4 LA 80 300
5 San-Diego 110 260
6 San-Bernadino 110 300
7 Albequerque 330 280
8 Kansas-City 620 360
9 Tucson 240 220
10 El-Paso 350 190
11 San-Antonio 540 110
12 Houston 630 130
13 New-Orleans 730 120
14 Fargo 570 580
15 Chicago 750 460
16 Salt-Lake-City 270 440
17 Denver 430 400
18 Dallas 560 210
19 Raleigh 980 320
20 Cleavland 890 460
21 Washington-DC 990 410
22 Savannah 940 200
23 Orlando 950 90
24 Miami 980 40
25 New-York 1040 480
26 Boston 1090 530
27 Montreal 990 610
```

Rails:



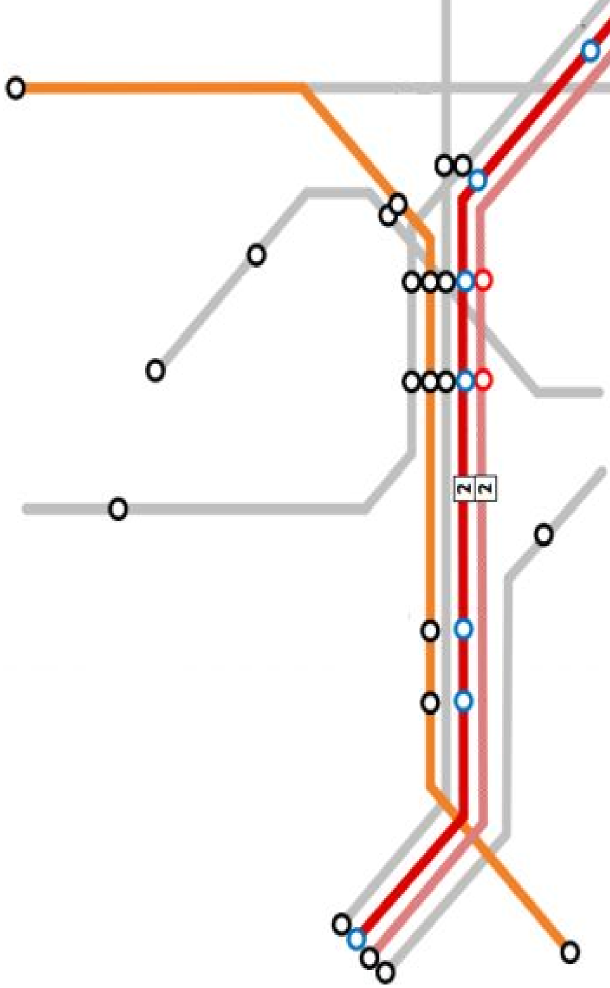
```
32
0 1 -1
0 14 -1
14 15 -1
15 20 -1
20 21 -1
21 25 -1
25 26 -1
25 27 -1
1 2 -1
2 3 -1
2 16 -1
16 17 -1
17 15 -1
17 15 -1
3 4 -1
4 5 -1
4 6 -1
4 9 -1
6 7 -1
7 8 -1
8 15 -1
9 10 -1
10 11 -1
11 12 -1
11 18 -1
18 15 -1
12 13 -1
13 19 -1
13 15 -1
19 22 -1
22 23 -1
23 24 -1
19 21 -1
```

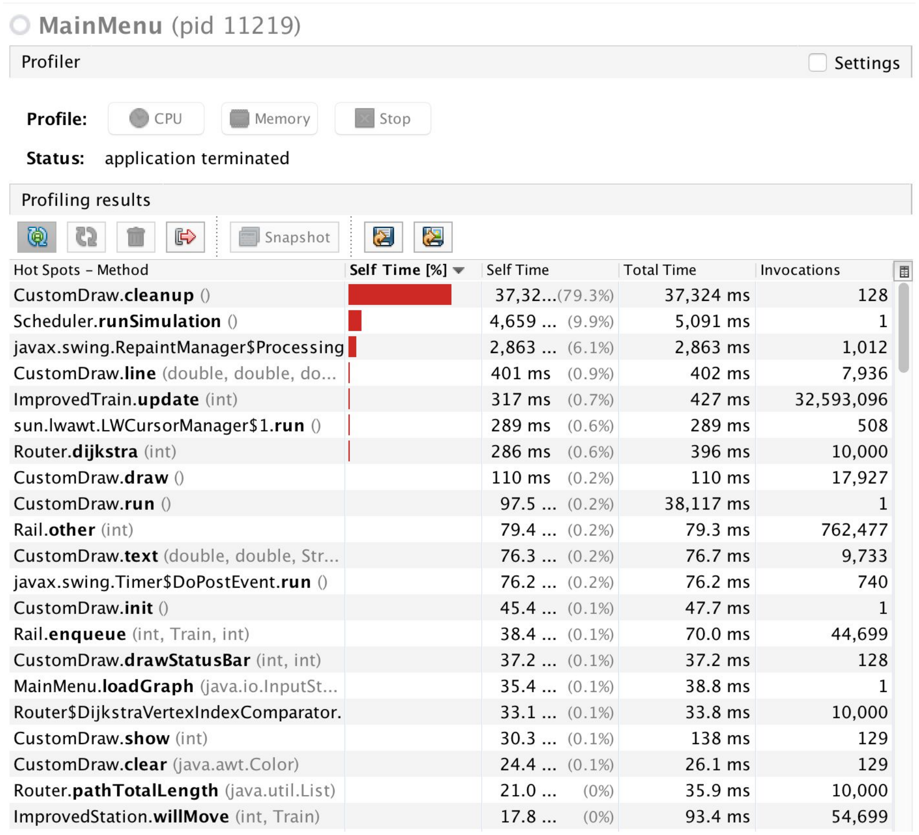


IMPLEMENTATION OF THE SOLUTION

Both the baseline and the improved train network use dijkstra’s algorithm to go from their source to their destination on the shortest path

- The difference is in how severely the trains are delayed





MainMenu (pid 11219)

Profiler Settings

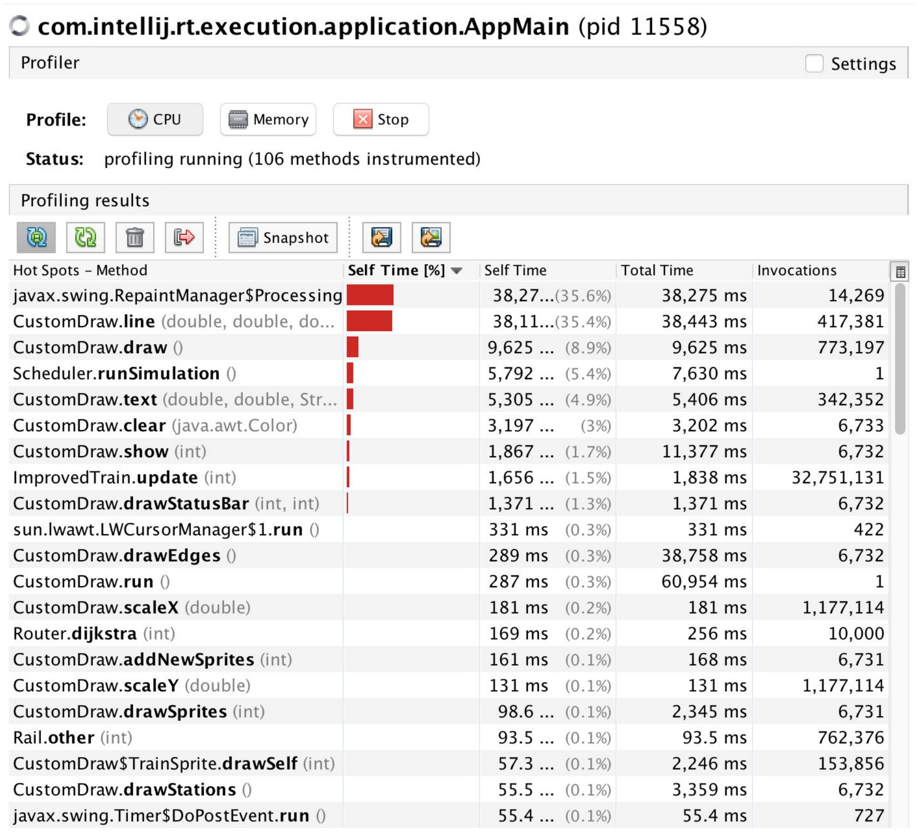
Profile: CPU Memory Stop

Status: application terminated

Profiling results

Snapshot

Hot Spots – Method	Self Time [%]	Self Time	Total Time	Invocations
CustomDraw.**cleanup** ()		37,32...(79.3%)	37,324 ms	128
Scheduler.**runSimulation** ()		4,659 ... (9.9%)	5,091 ms	1
javax.swing.RepaintManager$Processing		2,863 ... (6.1%)	2,863 ms	1,012
CustomDraw.**line** (double, double, do...		401 ms (0.9%)	402 ms	7,936
ImprovedTrain.**update** (int)		317 ms (0.7%)	427 ms	32,593,096
sun.lwawt.LWCursorManager$1.**run** ()		289 ms (0.6%)	289 ms	508
Router.**dijkstra** (int)		286 ms (0.6%)	396 ms	10,000
CustomDraw.**draw** ()		110 ms (0.2%)	110 ms	17,927
CustomDraw.**run** ()		97.5 ... (0.2%)	38,117 ms	1
Rail.**other** (int)		79.4 ... (0.2%)	79.3 ms	762,477
CustomDraw.**text** (double, double, Str...		76.3 ... (0.2%)	76.7 ms	9,733
javax.swing.Timer$DoPostEvent.**run** ()		76.2 ... (0.2%)	76.2 ms	740
CustomDraw.**init** ()		45.4 ... (0.1%)	47.7 ms	1
Rail.**enqueue** (int, Train, int)		38.4 ... (0.1%)	70.0 ms	44,699
CustomDraw.**drawStatusBar** (int, int)		37.2 ... (0.1%)	37.2 ms	128
MainMenu.**loadGraph** (java.io.InputSt...		35.4 ... (0.1%)	38.8 ms	1
Router$DijkstraVertexIndexComparator.		33.1 ... (0.1%)	33.8 ms	10,000
CustomDraw.**show** (int)		30.3 ... (0.1%)	138 ms	129
CustomDraw.**clear** (java.awt.Color)		24.4 ... (0.1%)	26.1 ms	129
Router.**pathTotalLength** (java.util.List)		21.0 ... (0%)	35.9 ms	10,000
ImprovedStation.**willMove** (int, Train)		17.8 ... (0%)	93.4 ms	54,699



com.intellij.rt.execution.application.AppMain (pid 11558)

Profiler Settings

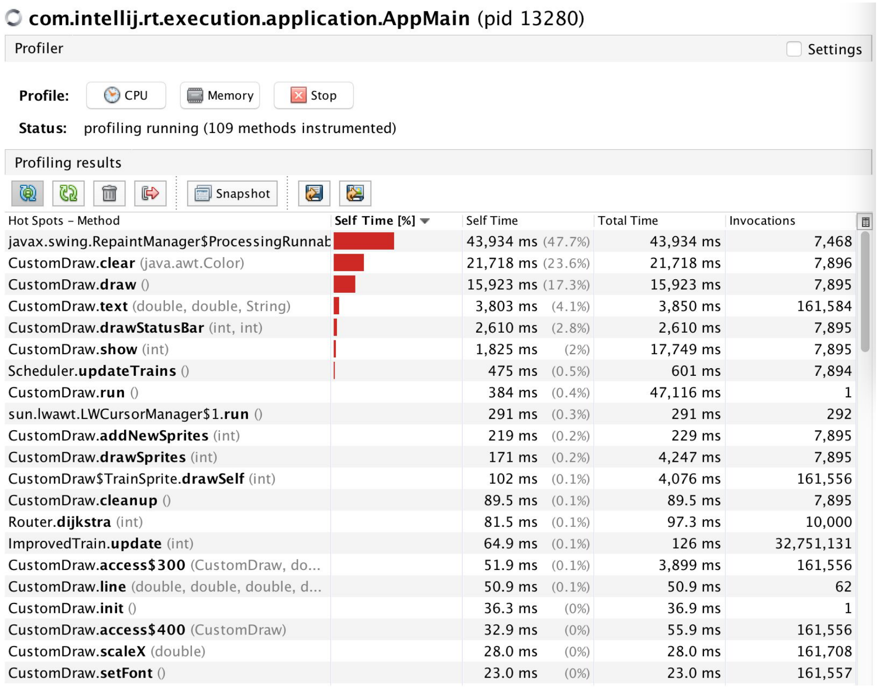
Profile: CPU Memory Stop

Status: profiling running (106 methods instrumented)

Profiling results

Snapshot

Hot Spots – Method	Self Time [%]	Self Time	Total Time	Invocations
javax.swing.RepaintManager$Processing		38,27...(35.6%)	38,275 ms	14,269
CustomDraw.**line** (double, double, do...		38,11...(35.4%)	38,443 ms	417,381
CustomDraw.**draw** ()		9,625 ... (8.9%)	9,625 ms	773,197
Scheduler.**runSimulation** ()		5,792 ... (5.4%)	7,630 ms	1
CustomDraw.**text** (double, double, Str...		5,305 ... (4.9%)	5,406 ms	342,352
CustomDraw.**clear** (java.awt.Color)		3,197 ... (3%)	3,202 ms	6,733
CustomDraw.**show** (int)		1,867 ... (1.7%)	11,377 ms	6,732
ImprovedTrain.**update** (int)		1,656 ... (1.5%)	1,838 ms	32,751,131
CustomDraw.**drawStatusBar** (int, int)		1,371 ... (1.3%)	1,371 ms	6,732
sun.lwawt.LWCursorManager$1.**run** ()		331 ms (0.3%)	331 ms	422
CustomDraw.**drawEdges** ()		289 ms (0.3%)	38,758 ms	6,732
CustomDraw.**run** ()		287 ms (0.3%)	60,954 ms	1
CustomDraw.**scaleX** (double)		181 ms (0.2%)	181 ms	1,177,114
Router.**dijkstra** (int)		169 ms (0.2%)	256 ms	10,000
CustomDraw.**addNewSprites** (int)		161 ms (0.1%)	168 ms	6,731
CustomDraw.**scaleY** (double)		131 ms (0.1%)	131 ms	1,177,114
CustomDraw.**drawSprites** (int)		98.6 ... (0.1%)	2,345 ms	6,731
Rail.**other** (int)		93.5 ... (0.1%)	93.5 ms	762,376
CustomDraw$TrainSprite.**drawSelf** (int)		57.3 ... (0.1%)	2,246 ms	153,856
CustomDraw.**drawStations** ()		55.5 ... (0.1%)	3,359 ms	6,732
javax.swing.Timer$DoPostEvent.**run** ()		55.4 ... (0.1%)	55.4 ms	727



com.intellij.rt.execution.application.AppMain (pid 13280)

Profiler — Settings

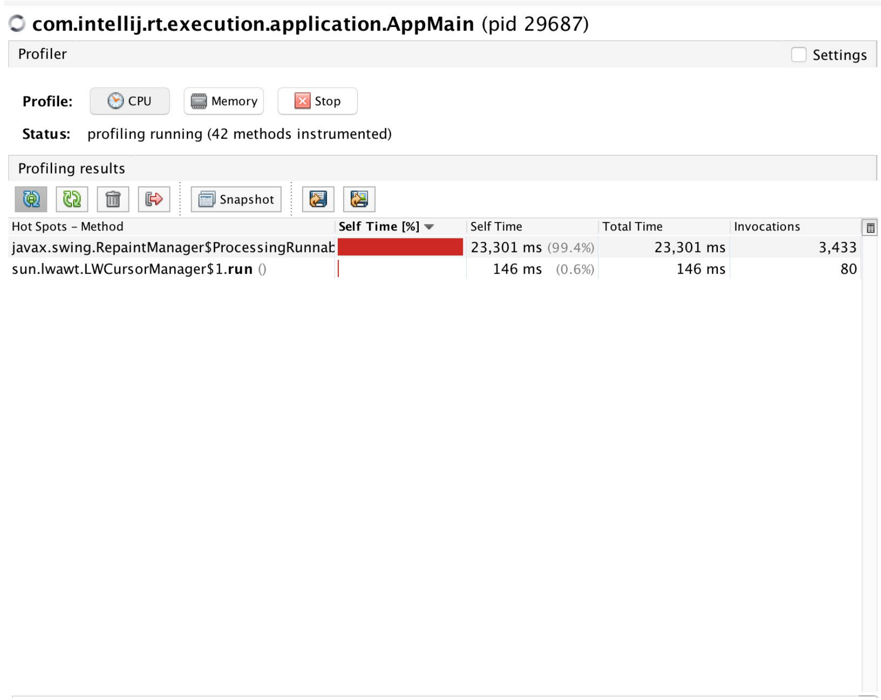
Profile: CPU Memory Stop

Status: profiling running (109 methods instrumented)

Profiling results

Snapshot

Hot Spots – Method	Self Time [%]	Self Time	Total Time	Invocations
javax.swing.RepaintManager$ProcessingRunnab	🟥	43,934 ms (47.7%)	43,934 ms	7,468
CustomDraw.**clear** (java.awt.Color)	🟥	21,718 ms (23.6%)	21,718 ms	7,896
CustomDraw.**draw** ()	🟥	15,923 ms (17.3%)	15,923 ms	7,895
CustomDraw.**text** (double, double, String)	🟥	3,803 ms (4.1%)	3,850 ms	161,584
CustomDraw.**drawStatusBar** (int, int)	🟥	2,610 ms (2.8%)	2,610 ms	7,895
CustomDraw.**show** (int)	🟥	1,825 ms (2%)	17,749 ms	7,895
Scheduler.**updateTrains** ()	🟥	475 ms (0.5%)	601 ms	7,894
CustomDraw.**run** ()		384 ms (0.4%)	47,116 ms	1
sun.lwawt.LWCursorManager$1.**run** ()		291 ms (0.3%)	291 ms	292
CustomDraw.**addNewSprites** (int)		219 ms (0.2%)	229 ms	7,895
CustomDraw.**drawSprites** (int)		171 ms (0.2%)	4,247 ms	7,895
CustomDraw$TrainSprite.**drawSelf** (int)		102 ms (0.1%)	4,076 ms	161,556
CustomDraw.**cleanup** ()		89.5 ms (0.1%)	89.5 ms	7,895
Router.**dijkstra** (int)		81.5 ms (0.1%)	97.3 ms	10,000
ImprovedTrain.**update** (int)		64.9 ms (0.1%)	126 ms	32,751,131
CustomDraw.**access$300** (CustomDraw, do...		51.9 ms (0.1%)	3,899 ms	161,556
CustomDraw.**line** (double, double, double, d...		50.9 ms (0.1%)	50.9 ms	62
CustomDraw.**init** ()		36.3 ms (0%)	36.9 ms	1
CustomDraw.**access$400** (CustomDraw)		32.9 ms (0%)	55.9 ms	161,556
CustomDraw.**scaleX** (double)		28.0 ms (0%)	28.0 ms	161,708
CustomDraw.**setFont** ()		23.0 ms (0%)	23.0 ms	161,557



com.intellij.rt.execution.application.AppMain (pid 29687)

Profiler — Settings

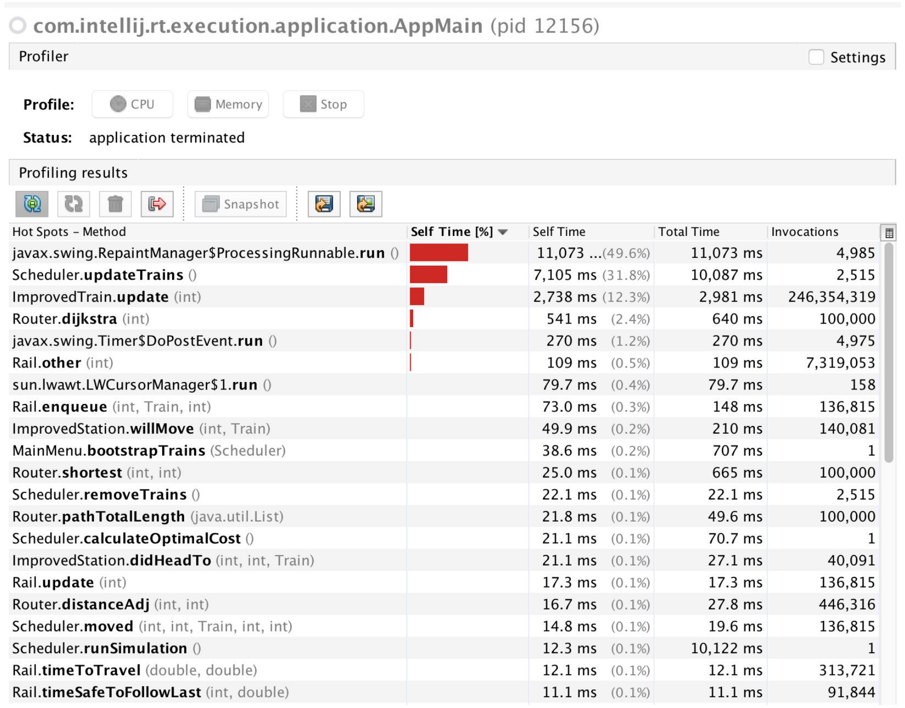
Profile: CPU Memory Stop

Status: profiling running (42 methods instrumented)

Profiling results

Snapshot

Hot Spots – Method	Self Time [%]	Self Time	Total Time	Invocations
javax.swing.RepaintManager$ProcessingRunnab	🟥	23,301 ms (99.4%)	23,301 ms	3,433
sun.lwawt.LWCursorManager$1.**run** ()	🟥	146 ms (0.6%)	146 ms	80



com.intellij.rt.execution.application.AppMain (pid 12156)

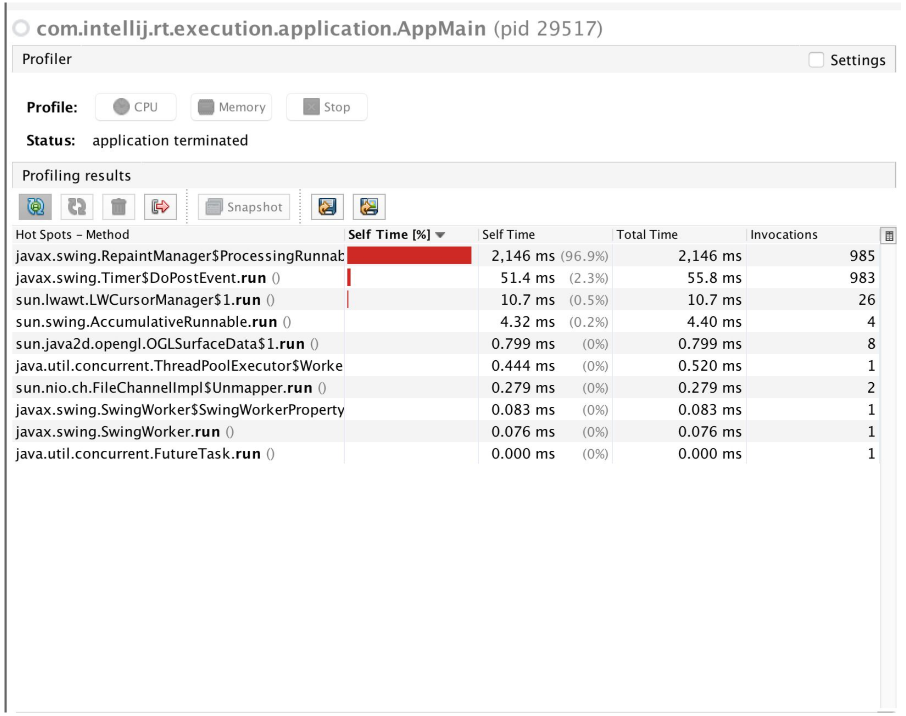
Profiler

Profile: CPU Memory Stop

Status: application terminated

Profiling results

Hot Spots – Method	Self Time [%]	Self Time	Total Time	Invocations
javax.swing.RepaintManager$ProcessingRunnable.**run** ()		11,073 ...(49.6%)	11,073 ms	4,985
Scheduler.**updateTrains** ()		7,105 ms (31.8%)	10,087 ms	2,515
ImprovedTrain.**update** (int)		2,738 ms (12.3%)	2,981 ms	246,354,319
Router.**dijkstra** (int)		541 ms (2.4%)	640 ms	100,000
javax.swing.Timer$DoPostEvent.**run** ()		270 ms (1.2%)	270 ms	4,975
Rail.**other** (int)		109 ms (0.5%)	109 ms	7,319,053
sun.lwawt.LWCursorManager$1.**run** ()		79.7 ms (0.4%)	79.7 ms	158
Rail.**enqueue** (int, Train, int)		73.0 ms (0.3%)	148 ms	136,815
ImprovedStation.**willMove** (int, Train)		49.9 ms (0.2%)	210 ms	140,081
MainMenu.**bootstrapTrains** (Scheduler)		38.6 ms (0.2%)	707 ms	1
Router.**shortest** (int, int)		25.0 ms (0.1%)	665 ms	100,000
Scheduler.**removeTrains** ()		22.1 ms (0.1%)	22.1 ms	2,515
Router.**pathTotalLength** (java.util.List)		21.8 ms (0.1%)	49.6 ms	100,000
Scheduler.**calculateOptimalCost** ()		21.1 ms (0.1%)	70.7 ms	1
ImprovedStation.**didHeadTo** (int, int, Train)		21.1 ms (0.1%)	27.1 ms	40,091
Rail.**update** (int)		17.3 ms (0.1%)	17.3 ms	136,815
Router.**distanceAdj** (int, int)		16.7 ms (0.1%)	27.8 ms	446,316
Scheduler.**moved** (int, int, Train, int, int)		14.8 ms (0.1%)	19.6 ms	136,815
Scheduler.**runSimulation** ()		12.3 ms (0.1%)	10,122 ms	1
Rail.**timeToTravel** (double, double)		12.1 ms (0.1%)	12.1 ms	313,721
Rail.**timeSafeToFollowLast** (int, double)		11.1 ms (0.1%)	11.1 ms	91,844



com.intellij.rt.execution.application.AppMain (pid 29517)

Profiler

Profile: CPU Memory Stop

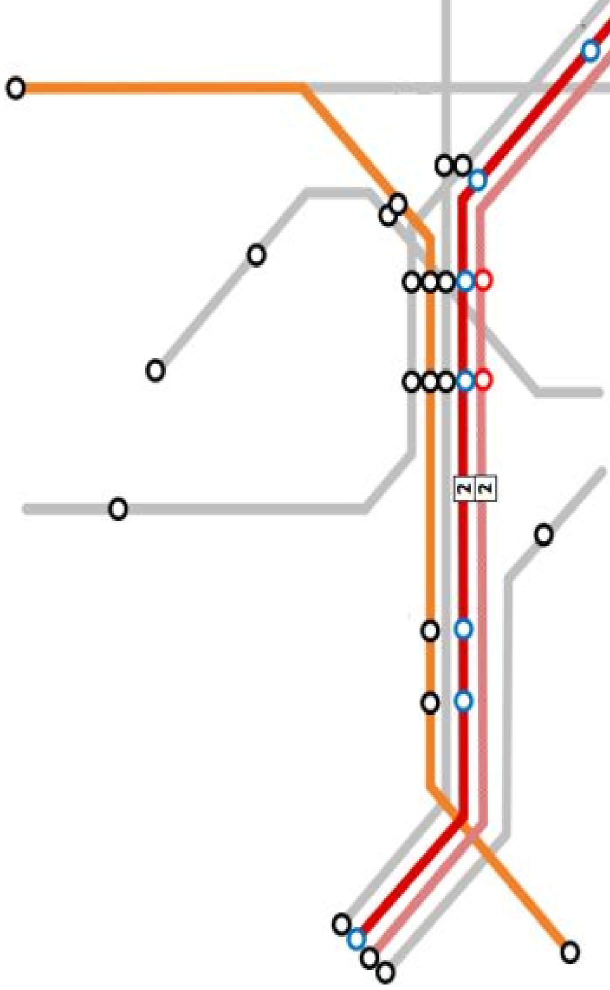
Status: application terminated

Profiling results

Hot Spots – Method	Self Time [%]	Self Time	Total Time	Invocations
javax.swing.RepaintManager$ProcessingRunnab		2,146 ms (96.9%)	2,146 ms	985
javax.swing.Timer$DoPostEvent.**run** ()		51.4 ms (2.3%)	55.8 ms	983
sun.lwawt.LWCursorManager$1.**run** ()		10.7 ms (0.5%)	10.7 ms	26
sun.swing.AccumulativeRunnable.**run** ()		4.32 ms (0.2%)	4.40 ms	4
sun.java2d.opengl.OGLSurfaceData$1.**run** ()		0.799 ms (0%)	0.799 ms	8
java.util.concurrent.ThreadPoolExecutor$Worke		0.444 ms (0%)	0.520 ms	1
sun.nio.ch.FileChannelImpl$Unmapper.**run** ()		0.279 ms (0%)	0.279 ms	2
javax.swing.SwingWorker$SwingWorkerProperty		0.083 ms (0%)	0.083 ms	1
javax.swing.SwingWorker.**run** ()		0.076 ms (0%)	0.076 ms	1
java.util.concurrent.FutureTask.**run** ()		0.000 ms (0%)	0.000 ms	1

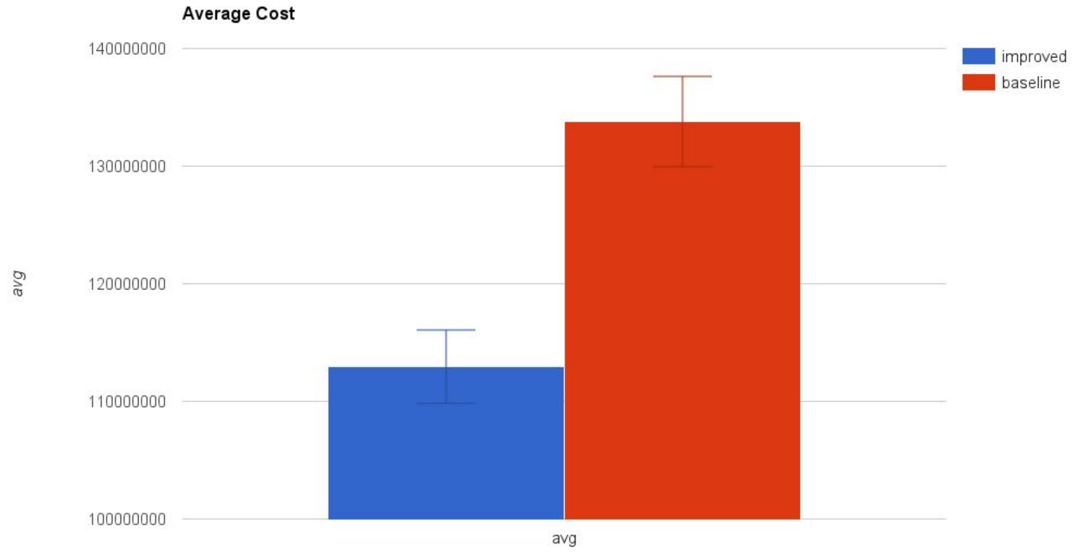
IMPROVEMENTS TO BE MADE

1. To take into account the breakdown or maintenance of railways and reroute trains during this circumstance
2. To create a two way rail system so that trains going in different directions can bypass each other at the same time.



RESULTS

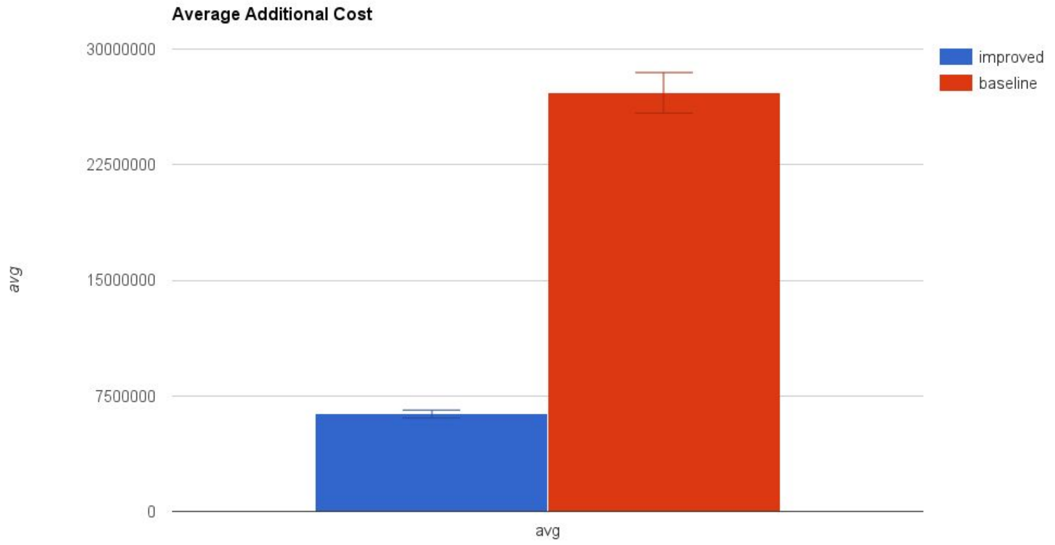
The average cost of the improved train network is about 84% cheaper



RESULTS

The additional cost factors in the cost for mileage

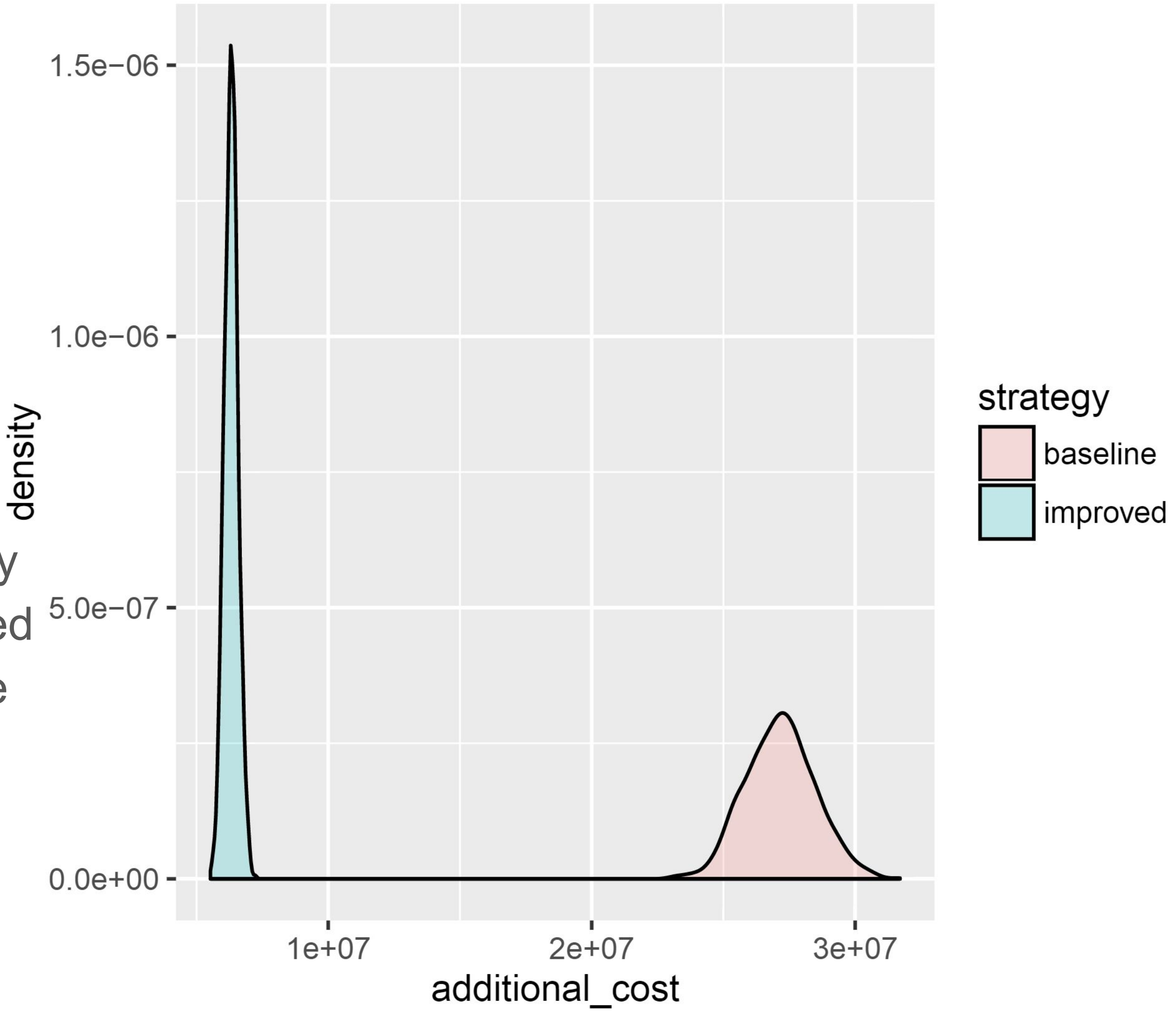
The improved cost is less than 1/4 of the baseline cost.



RESULTS

Estimated probability distribution: how likely the two costs will fall into the which range

The improved is more tightly distributed and is guaranteed to be cheaper than baseline


1.5e−06
1.0e−06
5.0e−07
0.0e+00
density
1e+07
2e+07
3e+07
additional_cost
strategy
baseline
improved