Validation of the results of Kaspi and Raviv (2013) on the Cross-Entropy heuristic for the integrated line planning and timetabling problem.

Bachelor Thesis Econometrics and Operations Research

Author: L.S Rook
Studentnumber: 333478
Supervisor: Dr. D.Huisman

June 30, 2014
Erasmus School of Economics,
Erasmus University Rotterdam

Abstract

Kaspi and Raviv (2013) were the first to successfully implement the cross-entropy meta-heuristic for the integrated line planning and timetabling problem, with the objective of minimizing both total operational costs and the user inconvenience, which is expressed by the total travel time of all passengers. In this thesis, their results are successfully verified by implementing and testing the CE algorithm in Java.

Contents

1 Introduction

Every day a lot of people travel by train. Long travel and waiting times, and a lot of transfers make passengers dissatisfied. Most passengers prefer a fast and direct connection from their origin to their destination. Obviously, it is not possible to fulfill this in an efficient manner for all passengers. To keep passengers satisfied and to stimulate them to travel by train more often, it is important to provide a good line planning and timetable which minimizes the total time passengers spend in the railway system.

Nowadays, the line planning and timetabling problem becomes more and more complex, because the railway systems extends over time. The line planning and timetabling problem are the two first problems in the planning problem process. The planning problem can be divided into different sub problems, namely: Line planning, timetabling, platforming, rolling stock circulation and crew rostering. In most previous studies these sub-problems are solved separately. The input for the next sub-problem is the output of the previous sub-problem. Kaspi and Raviv (2013) are the first, who solved the line planning and timetabling problem integrated through a Cross-Entropy (CE) metaheuristic introduced by Rubinstein (1999). The line planning problem consists of planning the lines that need to be covered and their frequencies. The timetabling problem uses the line planning to determine the arrival and departure time at each block and station.

In this thesis a CE algorithm for the integrated line planning and timetabling problem is implemented, with the purpose of minimizing the operational costs and the total travel time of the passengers. Operational costs are the total costs trains make from the beginning of their route until the end of their route. The total travel time of passengers starts when the passenger arrives at the departing station and involves all time until arrival on the final destination. So, the total travel time involves the total time spent in the train, transfer times and waiting times at stations. In this implementation a line plan and a cyclic timetable will be created. A cyclic timetable means a timetable for one cycle (one hour) after which this cycle is repeated over the planning horizon (one day). The departure times are the same every cycle, which makes it easier for the passengers to remember. It is not only easier for the passengers, but also for the planners because it is sufficient to plan a single cycle. This implementation is based on the CE algorithm as develop by, and tested on the instances used in Kaspi and Raviv (2013). For ease of use this thesis will replicate certain parts, and follow the notation, of the paper from Kaspi and Raviv (2013).

The available data is data about the Israëli railways in 2008. This data consist of information about the infrastructure and about the passenger demand per hour. The rail

passenger transport covers about 3% of the passenger transportation sector in Israël. The Israëli railway consist of approximately 1000 kilometers rail. The main connections are the connections near the coast and between Tel Aviv and Jeruzalem. Most trains depart every hour, some more frequently used trains depart multiple times per hour. There are no railway connections to neighbor countries and even between some large cities a railway connection is not provided. In the present line plan and timetable it is possible that traveling by bus is faster than traveling by train between some cities. This is another reason to create a line plan and timetable which reduces the total time spent in the railway system.

An explanation on definitions, modeling assumptions and constraints is given in Section 2 Problem description. In Section 3 an explanation of the CE heuristic is provided. The results can be found in Section 4 and the conclusion and suggestion for further research in Sections 5 and 6.

2 Problem description

In this chapter an explanation on definitions, modeling assumptions and constraints for the integrated line planning and timetabling problem is given.
Kaspi and Raviv (2013) give the following problem definition:
" The service-oriented line planning and timetabling problem is defined as follows: given a pool of routes, passenger demand for journeys, cycle time, horizon time, and safety and operational restrictions find a line plan and a cyclic timetable that together minimize the total cost associated with both the travel time of all passengers and the operation of all trains."

The line planning problem consist of planning the lines that need to be covered and their frequencies. The timetabling problem uses the line planning to determine the arrival and departure time at each block and station. Therefore, the output contains a list of all scheduled trains in one cycle. Each scheduled train consists of a route along with a set of stopping stations to wait, to let another train pass or to let the passengers get on and off the train, the enter and exit times for each block and station along the route are also provided. To provide a cyclic timetable the scheduled trains in one cycle are repeated over the planning horizon.

2.1 Definitions

The railway network consist of track sections, and track sections consist of several blocks. A block is a small part of the track, between two stations or between a station and a signal. In each block at most one train at the same time is allowed. A signal is a traffic light on the railway, it is used for train drivers so that they no whether they must stop (red signal), slow down (yellow signal), or can proceed (green signal). Signals are used to divide large track sections into smaller blocks. A sequence of one or more parallel block is called a segment. There are also special blocks that are used to let other trains pass or wait, these special blocks are called sidings. The time it takes for the train to get through a block is defined as the traversing time.

Stations consists of multiple parallel and crossing track sections and is used for trains to cross over, to pass other trains or to let passengers on/off the train. There are two kind of stations: Passenger stations and Operational stations. A station is called a passenger station if passengers can get on and off the train. An operational station is only used for crossing or catching up of other trains. Between two major stations there are multiple blocks, sidings and stations. This is called a route.

2.2 Parameters

The infrastructure of the Israëli railway network consists of approximately 1000 kilometers of rail, 47 passenger stations and 30 operational stations and 28 possible routes. The set of all possible routes is called the route pool. Where each route is characterized by a direction (inbound/outbound), a frequency, a priority, and a sequence of blocks and stations. Each block is characterized by a block id number and a traversing time and each station by a station id number and a capacity. The cycle time (C) has a length of 60 minutes and the planning horizon (T) is 1140 minutes. Since we make a timetable for one hour and replicate this over the whole planning horizon, there are 19 hours in a day in which passengers can travel by trains, so one cycle is repeated 19 times. For each of the 19 hours the demand is different. Therefore, there are 19 O-D matrices available. We will use all the 19 O-D matrices as the demand of all passengers, to calculate the total journey time of all passengers during a day.

2.3 Constraints

It is important to make sure that passengers can travel safely from their origin to their destination, partly for this purpose there are safety and operational constraints. If the safety and operational constraints are satisfied, then the solution is a feasible solution.

2.3.1 Safety constraints

- **Block seizing :** To prevent collision, there is only one train allowed at the same time in each block.
- **Block headway ($\mathbf{H}_b$):** For trains in opposite direction, there need to be enough time to change blocks safely. The H_b is the extra time, after the previous train leaves block b ,for the next train to enter block b.
- **Station headway ($\mathbf{H}_s$):** For safety matters station s is seen as occupied for some extra time after a dwelled train left this station. Another train can only enter station s H_s minutes after the previous train left station s.

2.3.2 Operational constraints

- **Block utilization ($\mathbf{U}_b$):** The total amount of time block b can be occupied by trains during one cycle.
- **Station capacity ($\mathbf{V}_s$):** Maximum number of trains that can dwell at stations s at the same time.
- **Minimal dwelling time ($\mathbf{W}_s$):** Minimal stopping time at stations s to let passengers on and off the train.

- **Minimal transfer time ($\mathbf{R}_s$):** Minimal time passengers need to make a transfer at station s.

Further, there is a restriction on the frequency of each route. The frequency of a route represent the maximum number of trains that can have this route in one cycle. Predetermined is that there can be at most four trains inserted in each route during one cycle. Hence, the maximal frequency of each route is four.

2.4 Assumptions

As in every model there are a few assumptions made to simplify the problem. Since we want to minimize the total travel time, we assume that each passenger wants to get to his destination as early as possible, in other words each passenger takes the fastest route or combination of routes from their origin to their destination. Other assumptions are the following:

1. The total demand for journeys is unaffected by the offered line plan and timetable;
2. The arrival times of the passengers at their origin station is unaffected by the actual timetable;
3. The capacity of the trains is not binding;
4. All trains are identical in their maximum speed.

2.5 Objective function

The objective of the model is to minimize the total travel time of all passengers and the total operational costs. The total travel time of the passengers starts when they arrive at their departing station and involves all time until arrival upon their final destination. The total travel time includes: the time spent traveling, waiting and/or transferring. The total operational costs are specified as the total costs from the start point to the end point of each route.

$$w \cdot \sum_i TC_i \quad + \sum_{t=0}^{T} \sum_{s_1,s_2} O_{s_1,s_2,t} F(s_1, s_2, t) \tag{1}$$

Where w the weight ratio, TC_i the total cost of train i and $F(s_1, s_2, t)$ the travel time of a passenger and $O_{s_1,s_2,t}$ the number of passengers that travels from station s_1 to station s_2 starting at time t .

The objective function is a weighted sum of the total travel time and the total operational costs as in Equation(1), where the weight of the total travel time is set to 1 and the weight of the total operational costs to three different values: 0, 100 and 200. If the weight of the total operational costs is set to 0, the model only minimize the total travel time. The higher the weight of the total operational costs, the more likely it is that there are less trains scheduled in the optimal solution.

2.6 Decision variables

For every possible train we need to make the decisions whether or not this train is in use (IU), and when this possible train is in use we need to determine at which stations this train stops (SS). We also need to decide on the earliest time in a cycle this train can start form the origin station, the gate time (GT). For each gene, we create a boolean decision variable for the in use status, for the stopping stations a boolean vector of the size of the number of stations on the route, whereas the gate time is modelled as a integer in the range of 0 to cycle time.

3 Cross-Entropy heuristic

Since the service oriented line planning and timetabling problem is complex, there is a heuristic needed to solve it. Note that a heuristic provides a solution that does not have to be the optimal solution. The heuristic used in this thesis is the Cross-Entropy (CE) heuristic. The CE heuristic has the following two phases:

1. Generate and evaluate a sample of random solutions according to a specified random mechanism;
2. Update the parameters of the random mechanism on the basis of these solutions in order to produce a better sample in the next iteration.

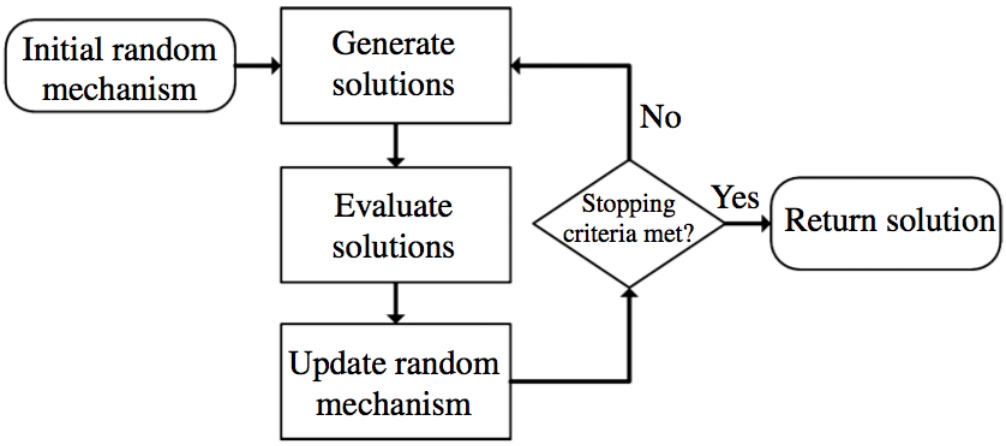

Figure 1: Outline of the CE algorithm, *Kaspi and Raviv (2013) Figure 3*

This section provides an explanation on the phases of the CE algorithm, as stated in Figure 1. It contains methods on: how to generate initial solutions based on the initial random mechanism, how to evaluate the solutions, how to update the parameters of the random mechanism to generate better solutions and how to set the stopping criteria.

3.1 Generate solutions

Generate solutions means that there must be as much solutions created as the size of the generation (GS), which is for us to set. We refer to a solution as a chromosome, thus every generation there must be GS chromosomes created. A chromosome is a list of genes, where a gene is the variable that needs to be optimized. In this thesis a gene represents a possible train, which corresponds to a route from the route pool.

Because a gene corresponds to a route, it has an id, direction and priority. As mentioned in Section 2.2 the id, direction, priority and frequency are given. The priority determines which routes are the most important and if two trains competing for the same block the train that corresponds to the route with the lowest priority is preferred. The frequency determines the maximum number of possible trains that can correspond to the same route each cycle. Since the frequency for all possible trains is 4 and there are 28 different routes, there are 112 genes in each chromosome. Further, each gene contains the following decision variables:

- In Use (IU): a boolean variable that indicates whether or not this possible train is used.
- Stopping Stations (SS): a boolean vector that indicates for each station in the route if this station is a stopping station or not.
- Gate Time (GT): an integer variable that indicates the earliest time in minutes a possible train can get into the first block of their route.

We will introduce a small example, which we will use as additional explaining of the steps in the next sections. This example consists of 3 trains (T1, T3, T5) which are all three in use and a cycle of 17 minutes. The route and gate times of the trains can be found in Figure 2. All the stations in the route of these three trains are stopping stations. T1,T3 and T5 are the trains, the route consists of blocks and stopping stations. This example uses the blocks: 1001, 1002, 1003 and 1004 and the stations: 2001, 2002, 2003 and 2004.

Route

Train 1	1001	2001	2002	1002	2003	2004	1003		GT: 2
Train 3	1001	2001	1002	2002	2003	1003	2004	1004	GT: 6
Train 5	1002	2002	1003	2003	2004	1004			GT: 0

Figure 2: Example routes.

3.1.1 Initial solutions

To start the CE algorithm we need a first initial generation. For that purpose, we generate the first GS chromosomes with default values for the separate distributions. Therefore, the first step in creating a line plan and cyclic timetable is to create initial solutions based on the initial random mechanism. With this initial random mechanism a generation of chromosomes is created. In the initial random mechanism the boolean variables IU and vector of boolean variables SS are sampled from a Bernoulli distribution with initial parameter

0.5 and the integer variable GT is initially sampled from a uniform distribution with cycle time, which means for every minute in the cycle time we have a separate distribution.

We need to keep track of a distribution set (IU,SS,GT) for every gene, note that for the initial solution these distribution sets are all the same because of the predetermined values of the initial parameters. For the implementation of the distribution sets we need to keep in mind that the distribution sets needs to be updated after each generation. How this updating works will be explained later on in Section 3.3.

3.2 Evaluate solution

The evaluation of each chromosome, contains four steps: Construct a line plan and cyclic timetable, make a graph of feasible itineraries, construct a reachability algorithm to determine the travel time from each event to each station and the calculation of the objective function value.

3.2.1 Construct a line plan and cyclic timetable

After creating chromosomes, we need to know if these chromosomes are feasible. A chromosome is feasible if all the genes that are in use (IU=1) can be inserted in all blocks and stopping stations along their route without violating any of the constraints. Inserting the genes is done by using a Greedy insertion algorithm. If the greedy insertion algorithm is completed successfully, the chromosome is a feasible solution, i.e all trains can be scheduled. We then have a line plan (all genes with IU = trains) and a cyclic timetable.

The greedy insertion algorithm tries to insert all genes with IU=1 in every block and stations along their predetermined route, important is that this is done in non increasing order of the given priorities. For every gene the time at which this train can enter his first block is set to the gate-time (GT), for every other block the time is set to the exit time of the previous block, if the previous block is a station then the enter time for the next block is set to the exit time of the block before the station plus the minimal dwelling time at the station. The enter time of stations equals the exit time of the block before the station. As explained in Section 3.1 each chromosome consist of 112 genes. The decision variables (IU, SS and GT) for each gene are randomly generated based on the random mechanism explained in Section 3.3. Further we provide every gene with a train id number related to the direction. The train id is odd for outbound trains and even for inbound trains.

At this point we already know which genes from a chromosome are in use, IU=1, therefore the first step is to sort all the genes that have IU=1 in non increasing order based on their priority. The greedy insertion algorithm determines if this gene can be inserted. A gene can only be inserted if non of the safety and operational constraints are violated.

The algorithm needs to keep track of the occupancy of all the blocks and stations. Therefore, a block occupancy array and a station occupancy array are made for every block and every station. The block occupancy array is an array of cycle time minutes in which the occupancy of this block is stored. This is done by inserting the train id at the minutes in which that train occupies this block. The station occupancy is an array of cycle time minutes where the capacity of this station at each minute is stored. In Figure 3 and 4 an example of four block occupancy arrays and four station occupancy arrays for the example as defined in Section 3.1 is provided. In Figure 3, we can see for every minute of the cycle time which block is occupied and if a block is occupied we can see which train occupies it. In between the braces after the block id is the traverse time of this block. In Figure 4, we can see what the station capacity of each station is for every minute. In between the braces in the station occupancy array we can see which trains are dwelling at that station. For example at time 8 there are 2 trains dwelling at station 1002, train T1 and train T3.

Block Occupancy

Block / minute	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
2001 (2)			T1	T1			T3	T3										
2002 (3)		T5	T5	T5	T1	T1	T1			T3	T3	T3						
2003 (2)						T5	T5		T1	T1			T3	T3				
2004 (1)								T5			T1					T3		

Figure 3: Example of four block occupancy arrays

Station Occupancy

Station cap/ minute	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1001		1(T1)	1 (T1)			1(T3)	1(T3)											
1002	1(T5)	1(T5)						1(T1)	2(T1 & T3)	1(T3)								
1003					1(T5)	1(T5)						1(T1)	1(T1)		1(T3)	1(T3)		
1004							1(T5)	1(T5)									1(T3)	1(T3)

Figure 4: Example of four station occupancy arrays

The first gene from the sorted list is taken and each element of the route that belongs to this gene is considered until a block or station cannot be inserted. When this happens, there are two options: First, this gene cannot be inserted and therefore this chromosome is infeasible. Whenever a chromosome is infeasible, we set the objective function value to a max-value. Second, this gene is set to non scheduled but the chromosome is left feasible and therefore the next gene from this chromosome is considered. Start with setting the time to the GT of this gene. Each element of the route can be a passenger station, a block or an operational station. The main focus is on the blocks, the stations are inserted based

on the enter times of the block following the station.

If the next element is a block, get the traverse time of this block. Search through the block occupancy array for the first free time interval of length *traverse time*, the start of this interval cannot be earlier than the GT for the first block or the exit time of the previous block for the other blocks. If such an interval is found, the block minutes are free. Insert this gene into this block by placing the train id at the minutes of the interval in the block occupancy array. Now these minutes for this block are occupied. If such an interval does not exist or if the block utilization is reached this gene cannot be inserted, since the decision variable IU=1 which basically means that it must be possible to insert this gene. For option 1 set this chromosome to infeasible. Create the next chromosome and start over. For the second option, set this gene to non scheduled and consider the next gene from the same chromosome.

The next step is looking at the previous element of the route. If the previous element is a station, we need to know when this gene/train dwelled at this station and whether or not the maximum station capacity at that time has been reached. Because we inserted this gene into this block, the enter- and exit-time of this gene from this block is known. Since this station is before the block, the time this gene enter this block (*enter time*) is the time at which this gene leaves the station. A train dwells at a station for W_s minutes plus the H_s. Search in the station occupancy array if the minutes from the *enter time block* - W_s until the *enter time block* + H_s have not reached the maximum station capacity. If the maximum station capacity has not been reached yet, this gene can be inserted in this station, i.e. this train can dwell at this station. Increase the station occupancy with one for the minutes this gene dwells at this station. If the maximum station capacity has been reached, this train cannot enter the station. This gene cannot be inserted in this station and for option 1 this chromosome is infeasible. Create the next chromosome and start over. For option 2, set this gene to non scheduled and start over with the next gene from the same chromosome.

If the previous element of the route is not a station, it must be a block. This previous block has already survived all these steps and thus we can find the enter- and exit-time of this previous block in the block occupancy array. (Otherwise this chromosome was infeasible and we had not got to this next block we are now). When there are multiple consecutive blocks this is called a sequence of blocks. In a sequence of blocks the exit time of the previous block must be equal to the enter time of the next block (the block we are examining now). So, if the enter time of this block is not equal to the exit time of the previous block we need to try to extend the time intervals the previous block occupies the block in the block occupancy array. Take the enter time of this block and search in the block occupancy array of the previous block if the minutes between the exit time of the previous block and the enter time of this block are free. If these minutes are free, extend

the previous block to the enter time of this block. If these minutes are not free, the previous block cannot be extended. For the first option, the chromosome is infeasible, because there is a gap between two consecutive blocks and the train cannot drive or dwell somewhere in the minutes of this gap. For the second option, set this gene to non scheduled and continue with the next gene from the same chromosome.

If the previous element is a block, we also need to take a look at the element after this block. If the next element is a station, then the sequence of blocks ends. Before inserting this gene into the next station, we first try to reschedule the sequence of blocks because the blocks should be traversed at maximum speed. The block enter time of the block in the sequence with the largest traverse time is fixed. The block exit time of this block is the enter time plus the traverse time. This exit time is fixed too. Try to reschedule all the other blocks in the sequence in a way that none of the other blocks must be extended and all blocks fit perfectly, i.e. the enter time of the next block equals the exit time of the previous block and all blocks in the sequence are traversed at maximum speed. Note that when it is necessary to reschedule the first block in the sequence to the left, then this train needs to dwell longer at the station before this first block. This must be possible in the sense that the maximum station capacity at the extra minutes must not be reached yet, if this is not possible rescheduling is not allowed and therefore the gene cannot be inserted and for the first option the chromosome is infeasible, and for the second option this gene is not scheduled but the chromosome is left feasible and therefore start over with the next gene.

If the next element is not a station, it is a block. This means that the sequence of blocks increase with one extra block. Nothing has to be done, the next block eventually reached the sequence of blocks routine.

We assume that if a station is not a stopping station, but the time a train dwells at the station is at least W_s. This station becomes a stopping station, because it is not efficient to dwell at a station for W_s minutes and not let passengers on and off the train. Therefore, we will investigate all non-stopping stations. For every non-stopping stations (SS_j=false) search if there are trains dwelling W_s minutes, set these stations to a stopping station (SS_j=true).

Another assumption we made is that when a train arrives at its final station, this train leaves this station W_s minutes later. The final station is occupied until H_s minutes after this train leaves. If we do not let this train leaving its final station, this train dwells at this station until the end of the cycle time. Which means that there is less station capacity for other trains from the moment this train enters the station until the end of the cycle. Therefore, we let the trains leave its final station. Note, leaving this station is not a departure event.

3.2.2 Graphical feasible itineraries

The timetable of a feasible solution is constructed in the previous step. So, we now know the arrival and departure times at all the stations for all the scheduled trains in this solution. The second step of the solution evaluation is creating an itineraries graph. An itineraries graph is needed for the calculation of the total travel time of all passengers and is based on the timetable obtained from the first step of the solution evaluation. The itineraries graph is a graph of all possible itineraries, with a node for every train event of the stations in the obtained timetable. It is a directed acyclic weighted graph with a maximum out degree of two. An out degree of two means that there are at most two connections starting at this node to neighbors. The maximum number of connections in a directed graph is $n(n-1)$, with n the number of nodes (train events).

To create an itineraries graph, we must create train events. There are two types of train events in the timetable: an arrival event and a departure event. Each train event contains the following information: a station id, a train id, arrival or departure type and a time. The creation of the list of train events is as following:
For every scheduled train, we need the blocklist with all the blocks along the route of this train. For the first station a departure event is created at the time at which the train enters his first block.

For all stopping stations of a gene: For the first station there is always a departure event at the exit time at this station + the cycle time. The cycle time, because we repeat the same line planning every cycle.

- If the *enter time of the station + the cycle time > the horizon time* :
  This station will not be reached, because the enter time exceed the horizon. Therefore, there is no event. Otherwise, create an arrival event at time: station enter time + cycle time.

- If the *exit time of the station < enter time of the station*:
  this means that during the dwelling time the train exceeds this cycle. Note that the exit time + the cycle time of this station needs to be increased with cycle time minutes.
  If this *new exit time* does not exceeds *the horizon time*:
  create a departure event at time: exit time of the station + cycle time + cycle time.

- If *the exit time of the station + cycle time* does not exceed *the horizon time*:
  create a departure event at time: exit time of the station + cycle time.

For the last station there is an arrival event only if the enter time of this station plus the cycle time does not exceed the horizon time. The time of the arrival event is the enter time

of the station + cycle time. For the last station there is no departure event.

Once all the train events are made, these train events needs to be connected. Every arrival event has a connection to the following two train events:

A1) The next arrival event of the same train in the next station;

A2) The first possible departure event of this station, which is at least W_s minutes later.

And each departure event has a connection to the following two train events:

D1) The next arrival event of the same train, in the next station;

D2) The next departure event from another train, in the same station.

Based on the example we introduced earlier, the timetable can be found in Figure 5 and in Figure 6 the itinerary graph is presented. On the horizontal axis the time in minutes from the cycle time and on the vertical axis the stopping stations. The green dots are the arrival events and the red-dots are the departure events, the connections are based on the specified connections described above: $A_1,\ A_2,\ D_1\ and\ D_2$. We put these next to the lines in the graph, only to show which of the four possible connection each line represents. The yellow lines represent the A_1 connection, the green lines the A_2 , the grey lines the D_1 and the black lines the D_2 connection. So, the $A_1,\ A_2,\ D_1\ and\ D_2$ in Figure 6 does not represent the length or weight of the connections!

Timetable

Station / Train	T1	T3	T5
1001	D 07.02	D 07.06	
1002	A 07.07 D 07.08	A 07.08 D 07.09	D 07.01
1003	A 07.11	A 07.14 D 07.15	A 07.04 D 07.05
1004		A 07.16	A 07.07

Figure 5: Example timetable

The length of the connections is the time differences between the two connecting nodes(train events). Once the itineraries graph is constructed a shortest path must be found. Every path is a feasible itinerary and the length of a path is the total travel time including transfer times. Further explanation can be found in Section 3.2.3.

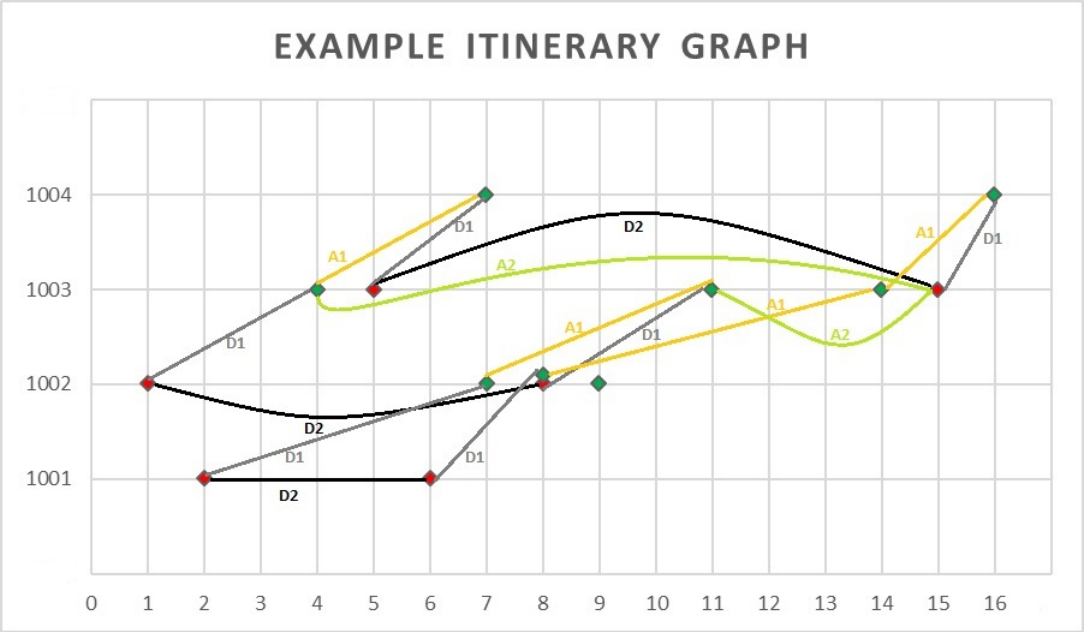

Figure 6: Example itinerary graph

3.2.3 Reachability algorithm

Once the itineraries graph is constructed, a shortest path must be found. Finding the shortest path from the departure event at the first station to the arrival event of the last station is equal to calculating the total travel time function $F(s_1, s_2, t)$. The reachability time is the earliest time that a station can be reached from a given train event and is therefore a shortest path problem. In the article of Kaspi and Raviv(2013) a few methods are presented. They name for example the Floyd Warshall algorithm and Johnson's algorithm. However, the itinerary graph is a directed acyclic graph with an out-degree of maximal two and only the earliest arrival event at each station is important. Because of these two properties a more efficient algorithm is used. For validation purposes, we will use the more efficient method presented in the article.

The used method consist of two steps:
First, sort the train events in non decreasing order of the time. Second, for each train event the reachability times need to be calculated. The earliest time for a train event is a integer vector for all stations, since we can travel from every station to every other station. Calculating the reachability times for each station is done by looking at the next arrival and next departure earliest time values and choosing the minimum of these two values. When we do this for all train events, the earliest time to the station vector for every train

event is always filled with the earliest travel time to the station. A special situation occurs when you cannot reach a station. In that case the earliest time value is set to the highest integer value available.

3.2.4 Calculation of the objective function

The objective function value consist of a weighted sum of the total operational costs for the usage of the trains and the total travel time of all the passengers. The costs for using a train are calculated as the time between the departure at the first station and the arrival at the last station of the route. To find the total costs we sum over all trains that are in use in the timetable. The total travel time of a flow of passengers is the time between arriving at the station from which they depart and the reachability time to the arrival event in the station of their destination and can be calculated by using (3) which is a combination of (2) and the second part of (1).

$$F(s_1, s_2, t) = ET(d, s_2) \ - \ t \tag{2}$$

$$\sum_{t=0}^{T} \sum_{s_1,s_2} O_{s_1,s_2,t}(ET(d, s_2) \ - \ t) \tag{3}$$

Where $ET(d, s_2)$ is the earliest time from d to station s_2 with d the first departure event at station s_1 following the passenger's arrival. Equation (3) is the result of filling equation (2) into equation (1). With equation (3) the total travel times of all passengers can be calculated. The objective function value can be calculated by Equation(4), where the $weight(ratio)$ can be set to 0, 100 or 200. TC is the total cost of using a train : time arrival last station - time departing first station, $i \in$ *all trains in use*. In words :

$$weight \cdot total\ costs \ + \ total\ travel\ time\ of\ all\ passengers$$

$$w \cdot \sum_{i} TC_i \ + \ \sum_{t=0}^{T} \sum_{s_1,s_2} O_{s_1,s_2,t}(ET(d, s_2) \ - \ t) \tag{4}$$

3.3 Update parameters random mechanism

After generating GS random chromosomes and evaluating these chromosomes, we select a certain (γ) percentage of the best feasible chromosomes based on the objective values. Since the goal is to minimize both travel time and costs, the best γ % solutions are the solutions with the lowest objective function values. These best solutions are called the elite group. The characteristics of the solutions in the elite group are used to update the

distribution sets from which another generation of chromosomes is made.

Create a distribution set for every gene, as explained in Section 3.1.1 for the initial solution these distribution sets are all the same. The distribution set contains the probability distribution functions for the boolean and integer decision variables for every gene. The IU Boolean and the SS list of Booleans are sampled from a bernoulli distribution and the integer GT is sampled from a discrete probability distribution function.

The parameters we need for the update are the following:

$\mu_{i,t}$	distribution parameter of IU of gene i at iteration t;
$\upsilon_{i,j,t}$	distribution parameter of SS of the jth stop of gene i at iteration t;
$\eta_{i,r,t}$	the probability of selecting GT=r for the train of gene i at iteration t;
$IU_{k,i,t}$	IU value of the ith gene of solution k at iteration t;
$SS_{k,i,j,t}$	SS value of the jth stop of the ith gene of solution k at iteration t;
$GT_{k,i,t}$	GT value of the ith gene of solution k at iteration t;

Where α is the smooting parameter which prevents early convergence of the algorithm to a local minimum. The smooting parameter is the weight of the current generation, where $(1-\alpha)$ is the weight of the previous generations. The parameters are updated by using Equations (5),(6) and (7).

$$\mu_{i,t} = \alpha \cdot \frac{\sum_{k \in elite} IU_{k,i,t}}{\gamma \cdot GS} + (1-\alpha) \cdot \mu_{i,t-1} \tag{5}$$

$$\upsilon_{i,j,t} = \alpha \cdot \frac{\sum_{k \in elite, IU_{k,i,t}=1} SS_{k,i,j,t}}{\sum_{k \in elite} IU_{k,i,t}} + (1-\alpha) \cdot \upsilon_{i,j,t-1} \tag{6}$$

$$\eta_{i,r,t} = \alpha \cdot \frac{|k : k \in \ elite, IU_{k,i,t} = 1, GT_{k,i,t} = r|}{\sum_{k \in elite} IU_{k,i,t}} + (1-\alpha) \cdot \eta_{i,r,t-1} \tag{7}$$

3.4 Stopping criteria

The algorithm stops when one of the following four criteria is met:

1. When for all genes the distribution parameter $\mu_{i,t}$ is smaller than ϵ or greater than 1-ϵ or if a gene is never scheduled in all the chromosomes. And if $\mu_{i,t}$ is greater than $1-\epsilon$, then the other distribution parameters $\upsilon_{i,j,t}$ and $\eta_{i,r,t}$ must be greater than 1-ϵ or smaller than ϵ.
   Where ϵ is a very small value.

2. When the objective function value converge, stop if there is no improvement in the objective function value during the last ζ iterations.

3. When a predetermined number of iterations is exceeded;

4. When a predetermined time limit is exceeded.

4 Results

In this section we will present our results. In Section 4.1 we will verify the statements and results of the tuning experiment done by Kaspi and Raviv (2013). Verifying the results against possible criticism by generating as much random solutions within six hours, in other words without updating the distribution parameters, is done in Section 4.2. In Section 4.3 more sensitivity of the objective function value is tested by varying the minimal dwelling time and the minimal transfer time.

The CE algorithm is implemented in Java and run on multiple computers from 1,86 GHz Intel Core 2 DUO processor to 2,0 GHz AMD Quad Core processor.

As mentioned in the Introduction and Section 2.2, the CE algorithm is tested with available data about the Israëli railways, the data is about the 28 possible routes in the Israëli timetable of 2008. The infrastructure of the Israëli railway network consist of 47 passenger stations, 30 operational stations and 28 possible routes.

Since our main goal is to verify the results of Kaspi and Raviv (2013), we will use the same set of parameters. They have used the following set of parameters to tune the model:

Horizon time:	1140 minutes;
Cycle time:	60 minutes;
Block Headway (H_b):	1 minute;
Station Headway (H_s):	1 minute;
Block utilization (U_b):	75 %;
Minimal dwelling time (W_s):	1 minute;
Minimal transfer time (R_s):	4 minutes.

The weight of the total costs varies over the values: 0, 100 and 200. Note that if we set the weight of the total costs to 0, this is equal to minimizing only the total travel time of all passengers. As mentioned in Section 2.5 when the weight of the total costs is higher, less trains will be scheduled. This makes sense, because when the weight factor of the costs is higher, the travel time is less important. More trains will increase the total costs and decrease the travel time of the passengers, more trains means that more passengers can be satisfied. When the total costs weight factor is higher than the satisfaction of the passengers, the objective function value will increase. Less trains will lower the total costs, but will lead to less satisfaction of passengers. Therefore three different weight ratio's are tested.

4.1 Tuning experiment

For the sensitivity analysis the generation size varies over the values 500 and 100, the smoothing parameter α over 0.3 and 0.7 and the algorithm is tested with keeping the elite group for the next generation and without keeping the elite group for the next generation.

Due to the lack of time, we cannot fully re-do all the $3 \cdot 2^3$ factorial experiments with four replications. Therefore, we will choose the best combination of the smoothing parameter, the generation size and the keep elite for every weight of the total costs from Kaspi and Raviv. In Table 1 the three combinations, one for each weight, which provided the best results by Kaspi and Raviv are shown.

Ratio	α	GS	Keep Elite	Objective function value (hours/day) Average	Objective function value (hours/day) Minimum	Objective function value (hours/day) Range	Trains per cycle Average	Number of generations Average
0	0.7	1000	Yes	99643,2	991,8	99125,6	40.25	320.75
100	0.3	1000	No	154182,3	659,4	153755	14,75	494,50
200	0.3	1000	No	185562,6	1501,7	184930,3	10,25	595,25

Table 1: The three chosen settings from Kaspi and Raviv , including results.

Because we want to verify, or at least getting as close to the results of Kaspi and Raviv (2013) we will also execute each run four times and present the average of the objective function value over these four replications. The average of the objective function value in presented in hours per day. Along with the average objective function value, the average number of scheduled trains per cycle and the average number of generations will be given.

There are two options for the insertion process.The first option is to get rid of the chromosome as soon as a gene with in use =1 cannot be inserted in all blocks and stations along its route and therefore the updating of the parameters is done only with the genes that are actually scheduled. The second option is to keep all the chromosomes in a generation, when a gene cannot be inserted we will set this gene to not-scheduled. In the second case, non of the chromosomes are infeasible but some genes of a chromosome are set to not scheduled and the in use is still 1. The updating of the parameters is now done by all genes with in use =1 whether or not the genes can be scheduled. The article of Kaspi and Raviv (2013) claims that it is significantly better to set the in use genes of a chromosome that cannot be inserted to non-scheduled than it is to set the whole chromosome to infeasible. Their reasoning is, when a chromosome is infeasible it can be that there are genes in this chromosome that can be inserted and even with the non scheduled genes give a lower objective function value than some other chromosomes. *" It seems that ignoring trains that could not be scheduled may prematurely disqualify too many trains out of the generated solutions"*. Therefore the second option leads to better results. Kaspi and Raviv present the results

of the second option only. We will verify this statement by running both options for the three best combinations as described in Table 1. As stopping criterium only the 6 hour time limit is used.

				Option 1				
Ratio	α	GS	Keep Elite	Objective function value (hours/day) Average	Objective function value (hours/day) Minimum	Objective function value (hours/day) Range	Trains per cycle Average	Number of generations Average
0	0,7	1000	Yes	102956	102524	919	47,25	674
100	0,3	1000	No	156353	155884	831	15	767,25
200	0,3	1000	No	188744	186730	3171	10,75	803,75

Table 2: Results CE algorithm: Option 1 skip chromosome

				Option 2				
Ratio	α	GS	Keep Elite	Objective function value (hours/day) Average	Objective function value (hours/day) Minimum	Objective function value (hours/day) Range	Trains per cycle Average	Number of generations Average
0	0,7	1000	Yes	99734	99145	887	40,5	751,25
100	0,3	1000	No	155191,3	154754	879	14,5	682,75
200	0,3	1000	No	187147,8	1863419	1042	10,25	625,5

Table 3: Results CE algorithm: Option 2 skip gene

The results of using option 1 can be found in Table 2, and the results of using option 2 in Table 3. First, we will compare the results of both options. The results of option 2 are significantly better than the results from option 1. The average objective function value for all three ratio's is significantly lower for the second option, and for ratio 0 the average trains per cycle is remarkably lower for the second option. We can conclude that the second option indeed leads to significantly better results, just as Kaspi and Raviv claim. Since this second option leads to significant better results, all further results in this thesis are based on the second option. Secondly, we will compare our results from the second option (Table 3) with the results from Kaspi and Raviv (Table 1). For the ratio 0 our average objective function value differs 0,09 % from the average objective function value presented by Kaspi and Raviv. For ratio 100 and 200 this is respectively 0,65% and 0,85%. The average number of trains per cycle is approximately equal and so we can drawn the conclusion that the results from Kaspi and Raviv are verified.

Furthermore, it make sense that the average objective function value increases and the average number of trains per cycle decreases when the ratio increases. On one hand, with a ratio of 0 the total costs are not taken into account, this is equivalent to minimizing the total travel time of all passengers only. When only the total passenger travel time is of interest it is likely that more trains will be needed, because we only want to satisfy as much passengers as we can and more trains can result in reducing the total travel time of passengers. On the other hand, with a ratio of 200 the focus is more on minimizing the total costs of using the trains, so there will be less trains scheduled. Less scheduled trains will lead to longer passenger travel times, therefore the total passenger travel time will increases.

4.2 Running CE without updating the parameters

As mentioned in Kaspi and Raviv (2013): *"One possible criticism on the CE algorithm may be that the improvement of the solutions stem merely from the fact that many random solutions were generated"*. Kaspi and Raviv prove that the CE algorithm performs better than some simple random approach. They compare the CE algorithm with a so called Brute Force Approach. Again, we will reproduce this test and verify the results of this comparison.

The brute force approach use the initial random mechanism to generate as much feasible solution as possible within a time limit of six hours. This is equivalent to running the CE algorithm without updating the parameters, from Equations (5)-(7) we can see that this is running the CE-algorithm with $\alpha = 0$. and only activating the stopping criteria of six hours.

We will test $\alpha = 0$ for the three best combinations of parameters from Table 1, again each of the tests is executed four times. From these four replications, the best (minimum) objective function value in hours/day is taken and compared with the minimum objective function value and to the average objective function value in hours/day from Table 3 in Section 4.1 and thus determine the difference.

Table 4 contains the result of comparing the CE algorithm of the second option, see Section 4.1, with the Brute Force Approach, running the CE algorithm with $\alpha = 0$. For ratio 0, the best objective function value of the CE algorithm is 29,16% and the average objective function value of the CE algorithm is 28,8% lower than the best objective function value of the Brute Force Approach. For the ratio 100 this is respectively 21,84 % and 21.25% and for ratio 200 22.74% and 22.63 %. Our results leads to the same conclusion and therefore verify the results of Kaspi and Raviv: The performance of the CE algorithm is significantly

Ratio	α	GS	Keep Elite	Brute force best objective function value (hours/day)	Difference from best CE value (%)	Difference from average CE (%)
0	0	1000	Yes	130249,5	29,16	28,80
100	0	1000	No	188560	21,84	21,25
200	0	1000	No	229915	22,74	22,63

Table 4: Results comparing CE (second option) with Brute Force Approach ($\alpha = 0$)

better than the performance of some random approach.

4.3 Varying the minimal dwelling time and the minimum transfer time

Additionally, we will run some more test to investigate the sensitivity of the algorithm further. For this purpose, two parameters: minimum dwelling time (W_s) and the minimum transfer time (R_s) are varied. We are interested in the effects of varying these parameters, ceteris paribus. Due to the lack of time, we only tested the R_s for ratio 100 and 200 and the W_s for ratio 100, with α=0,3, GS=1000, and not keeping the elite group.

4.3.1 Varying the minimal dwelling time

We are interested in the effects on the average objective function value and the average number of trains per cycle, when we change the minimal dwelling time from 1 minute to 2 minutes, ceteris paribus. Changing the minimal dwelling time from 1 minute to 2 minutes means that trains are dwelling longer at stations, therefore it is expected that less trains can be scheduled in comparison to the results for ratio 100 from Table 3 When less trains can be scheduled the total operational costs reduces, but the travel time will increase since less passengers can be satisfied.

Ratio	α	GS	Keep Elite	Objective function value (hours/day). Average	Objective function value (hours/day). Minimum	Objective function value (hours/day). Range	Trains per cycle. Average	Number of generations. Average
100	0.3	1000	No	170445	169745	1211	13,5	252,5

Table 5: Results $W_s = 2$, ceteris paribus

We did four replications with all stopping criteria activated, for the no-improvement criteria we choose ζ=30 generations because from Table 5 in Kaspi and Raviv (2013) we can see that the *% from best* is less than 1%. We noticed that all four runs terminated by the

stopping criterium of no-improvement after 30 generations. Table 5 shows the result of running the CE algorithm with W_s= 2 minutes, ceteris paribus.

From Table 5, we can see that the average number of trains per cycle is now 13,5 trains, compared to the 14,5 trains from Table 3 this is a 6,9% decrease. Whereas the average objective function value increases with approximately 10 %. This is probably due to higher total costs, since a longer minimum dwelling time at stations ensures that it take more time for trains to complete their route which in their case leads to an increase in passenger travel time.

4.3.2 Varying the minimum transfer time

Another sensitivity analyses is tested in the form of varying the minimum transfer time (R_s). The R_s is set to 4 minutes and we are interested in what happens to the average objective function value and the average number of trains per cycle if we increase and decrease the minimum transfer time with one minute. So, we will test with a R_s of 3 minutes, ceteris paribus, and a R_5 of 5 minutes, ceteris paribus. If the minimum transfer time decreases (increases), it is possible that the total travel time of the passengers reduces (increases) slightly because passengers can transfer to other trains one minute sooner (later). Since most passengers prefer less transfers to more transfers, we expect that the total travel time of a passenger will only reduce with a couple of minutes at most. We will expect a minor effect or not even an effect at all, because of the small changes in the transfer time.

As in Section 4.3.1, we did four replications with all stopping criteria activated, and for the no-improvement criteria we choose ζ=30 generations. Again, we noticed that all four runs terminated by the stopping criterium of no-improvement after 30 generations. Table 6 shows the result of running the CE algorithm with R_s= 3 minutes, ceteris paribus and Table 7 shows the results of running the CE algorithm with R_s=5 minutes, ceteris paribus.

Ratio	α	GS	Keep Elite	Objective function value (hours/day). Average	Objective function value (hours/day). Minimum	Objective function value (hours/day). Range	Trains per cycle. Average	Number of generations. Average
100	0.3	1000	No	155061,8	154491	906	14,5	272,25

Table 6: Results $R_s = 3$, ceteris paribus

If we compare the results in Tables 6 and 7 with the results for ratio 100 in Table 3, we can conclude that varying the minimum transfer time does have a significant effect on the average objective function value and on the average number of trains per cycle. The

Ratio	α	GS	Keep Elite	Objective function value (hours/day). Average	Objective function value (hours/day). Minimum	Objective function value (hours/day). Range	Trains per cycle. Average	Number of generations. Average
100	0.3	1000	No	155130,8	154658	880	14	285

Table 7: Results $R_s = 5$, ceteris paribus

objective function value is not sensitive for small changes in the minimum transfer time.

5 Conclusion

The purpose of this thesis was to verify the results presented in the article for the integrated line planning and timetabling problem by Kaspi and Raviv (2013). Kaspi and Raviv have shown that they succeeded in applying a CE algorithm for the integrated line planning and timetabling problem.

This thesis verifies the results of Kaspi and Raviv by implementing the same CE algorithm using a Java implementation. We have verified that the results of the tuning experiment are approximately the same as the results for the same ratio's and algorithm parameters obtained by Kaspi and Raviv. The insertion process was tested in two different options, where we showed and therefore verified that the second option is significantly better. We do not know exactly how the C implementation of Kaspi and Raviv is done, but we can conclude from the results of the average number of generations within a predetermined time limit that our Java implementation runs considerably faster.

Furthermore, we had verified the conclusion, drawn by Kaspi and Raviv, that the CE algorithm is significantly better than a random Brute Force Approach. Additionally, we have investigated the sensitivity of the objective function value by varying the minimum dwelling time and the minimum transfer time. This lead us to the conclusion that the objective function value is not very sensitive for a small change in the minimum dwelling time and not at all for small changes in the minimum transfer time.

6 Further research

Kaspi and Raviv already mention a few options for further research. We think it might be interesting to extend the model with a capacity per train. This would make the model more useful for applying the Dutch situation, because in the Dutch situation user inconvenience due to full trains during rush hours is a major complaint. Implementation seems to be rather easy by adding a decision variable for the capacity in each gene and keeping track embarking and disembarking passengers in every station. When this is done, a further study for using this model in the Dutch situation might be considered then.

References

Kaspi M, Raviv T (2013) Service-Oriented Line Planning and Timetabling Problem for Passenger Trains, *Transportation Science*, Vol. 47 (3): 295:311.

Rubinstein RY (1999), The cross-entropy method for combinatorial and continuous optimization. *Methodology and Comput. Appl. Probab.* 2(2):127-190.