Service-Oriented Line Planning and Timetabling for Passenger Trains

Mor Kaspi and Tal Raviv

"Researchers in mathematical optimization should grasp the currently available momentum and opportunities in the railway industry by not focusing too much on theoretical results, but by going for real-world applications of their models and techniques. The latter will lead to a win-win situation, both for the researchers and for the railway industry"

Caprara et al. (2006)

Abstract

An integrated line planning and timetabling model is formulated with the objective of minimizing both user inconvenience and operational costs. User inconvenience is modeled as the total time passengers spend in a railway system, including waiting at origin and transfer stations. The model is solved using a Cross-Entropy metaheuristic. The line plan and timetable of Israel Railways is used as a benchmark. Using the same amount of resources, the average journey time of passengers is reduced by 20%.

Keywords: railway, line planning, timetabling, cross-entropy metaheuristic

1. Introduction

Railway operations' planning has been practiced for almost two centuries. Over the years, this task has become increasingly complex as railway systems develop. In the last few decades, railway planners have been using decision support systems and optimization methods in order to improve the quality of their plans and to save effort. The planning process consists of several phases; the fundamental ones are line planning, timetabling, rolling stock circulation, platforming and crew scheduling. Traditionally, the planning process is done hierarchically such that the outcome of one phase serves as an input for the following. For surveys on various railway planning optimization problems, see Bussieck et al. (1997), Cordeau et al. (1998), Caprara et al. (2006), Törnquist (2006) and Cacchiani and Toth (2011).

This paper considers the planning of both the lines and the timetable in a regional or metropolitan train system with the goal of minimizing operational costs

and user inconvenience. Operational costs are modeled as the total engine time, and user inconvenience is modeled as the total passenger *journey time*. Journey time includes both the riding times and the waiting times at origin and transfer stations.

The line planning problem is defined as follows: Given railway infrastructure, the traveling time over each of its segments and the passenger demands for journeys determine which set of lines is used and their frequencies of use. The objective function is typically either cost-oriented or service-oriented. Operational costs are modeled as a function of the lines and their frequencies, and in some cases, with respect to the type and capacity of the train, see Claessens et al. (1998) and Goossens et al. (2004). User inconvenience is modeled by the number of nondirect passengers, Bussieck et al. (1996), the riding time, Borndörfer et al. (2007), or the riding time with additional penalty for transfers, Schöbel and Scholl (2006). For a recent survey on line planning, see Schöbel (2011).

Bussieck et al. (1996) and Schöbel (2011) argue that user inconvenience should be measured by the total time passengers spend in the railway system, i.e., journey time. However, because a timetable is required for the calculation of the journey time, this cannot be done during the line planning phase.

The Line Planning Problem can be formulated as a Mixed Integer Linear Program (MILP) and solved using methods such as branch and bound, see Bussieck et al. (1996) and Claessens et al. (1998), branch and cut, see Goossens et al. (2004), and column generation, see Schöbel and Scholl (2006) and Borndörfer et al. (2007).

The train timetabling problem is defined as follows: For a given set of lines and frequencies, determine the arrival and departure time at each block (track section) and station such that a set of safety constraints is satisfied. In regional systems, the rail infrastructure is a limited resource on which several lines compete. Consequently, the safety constraints are more complicated in comparison to other transportation systems in which either each line has a separate rail (such as underground systems) or the infrastructure is not binding at all (such as bus systems).

A common practice in timetabling of regional rail systems is to schedule the trains in a cyclic manner. In such timetables, each event is repeated every cycle, e.g., every hour. This approach is preferred by both passengers and planners. For the planners, *cyclic timetables* are easier to design because it is sufficient to plan a single

cycle. Serafini and Ukovich (1989) propose a general mathematical framework for scheduling periodic activities: the Periodic Event Scheduling Problem (PESP). Many of the studies on railway cyclic timetabling have adopted the PESP paradigm. See for example Odijk (1996), Lindner (2000), Liebchen and Möhring (2002) and Liebchen and Möhring (2007).

Liebchen and Möhring (2007) note that the routing of passenger flows is beyond the scope of the PESP. Because the problem studied in this paper includes routing of passengers, a new cyclic timetable construction heuristic is devised.

Operational costs and user inconvenience are largely affected by decisions made both in the line planning phase and the timetabling phase. Hence, simultaneously solving the two problems may be beneficial. Several previous studies have integrated some aspects of the two phases. Ceder and Israeli (1992) and Michaelis and Schöbel (2009) deal with bus systems in which the infrastructure is not binding; therefore, once the lines and their frequencies are determined, a feasible timetable can be derived directly. In Gorman (1998) and Borndörfer et al. (2007), the line planning model is supplemented with congestion constraints for each track section. While this approach may be enough to assure a feasible timetable for freight trains or a tram system, this approach is not suitable for regional railway systems. Lindner (2000) presents an integrated model that combines cost-oriented line planning and timetabling. His proposed solution method is to decompose the problem back to its two subproblems.

Evaluation of the service level provided by line plans and timetables requires the routing of passenger flows. Most of the line planning literature makes simplifying assumptions about the behavior of passengers. Many assume that the routing of a passenger over the network links is predetermined and not affected by the line plan. One example is the system split approach used by Bouma and Oltrogge (1994). The use of predetermined passenger paths is also customary in timetabling; see for example Wong et al. (2008). Some recent studies have included routing of passengers within the line planning model, see Schöbel and Scholl (2006), Borndörfer et al. (2007) and Schmidt and Schöbel (2010). However, minimization of the total journey time is not considered as a goal in these studies.

Some studies deal with the problem of finding an optimal itinerary given an origin, a destination, a timetable, and a set of criteria. The resulting itinerary is provided as a service to the passengers. See Müller-Hannemann et al. (2007) for a survey on this topic. Kinder (2008) uses this approach to improve a given timetables. However, because the line plan is predetermined in his study, the effect on the total journey time of passengers is limited.

No previous study on railway planning has integrated line planning, timetabling and routing decisions of the passengers. The contribution of this study is in presenting (a) a model that combines these three problems and (b) an algorithm to solve it.

The rest of the paper is organized as follows: In Section 2, the service-oriented line planning and timetabling problem is defined. In Section 3, some properties of the optimal solution of this problem are derived. These properties are used by the solution method presented in Section 4. The results of a numerical study that is based on actual data from the Israeli railway system are reported in Section 5. In Section 6, some possible extensions of the problem are discussed. Finally, a conclusion is given in Section 7.

2. Problem Definition

The service-oriented line planning and timetabling problem (SOLPTP) is defined as follows: Given a *pool of routes*, *passenger demand for journeys*, *cycle time* (C), *horizon time* (T), 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.

2.1. Input

The railway infrastructure is divided into atomic units called *blocks* and *stations*. A block represents a regular track section and is characterized by a minimal traversing time. A station is a location that consists of several parallel and intersecting track sections where trains can overtake or cross each other. It is called a *passenger station* if passengers can embark or disembark there and an *operational station* if it is destined solely to allow overtaking and crossing.

A sequence of blocks and stations between two major stations that may be traversed by a train is called a *route*. We note that the term *line* used in the railway planning literature is more general than *route*. While line defines a sequence of segments to be traversed by a train, a route is specifically defined in terms of blocks. Each route has a numerical value referred to as a *priority*. When two trains compete for the same resource, the one that travels on a route with a higher priority value is given the right of way. The set of all possible routes to choose from is referred as the *route pool*. Note that the physical structure of the railway network is not given explicitly as an input, but it may be derived from the route pool.

The passenger demand for journeys is given in the form of origin-destination (O-D) matrices, with one matrix for each period of the planning horizon. The $O_{s_1,s_2,t}$ element of the O-D matrix is the number of passengers that wish to travel from station s_1 to station s_2 starting at period t. A period is an interval during the planning horizon (e.g., one or ten minutes).

2.2. **Output**

A *scheduled train* is defined by a route, a set of stopping stations, and consistent entrance and exit times at each block and station along the route with respect to a given scheduling cycle (typically one hour). A solution of the problem is defined by a list of scheduled trains.

The cyclic timetable is constructed by repeating all scheduled trains along the planning horizon. Note that there may be more than one scheduled train that corresponds to a certain route. The number of scheduled trains per route constitutes its frequency.

We note that previous railway line planning literature studied problems of selecting a subset of routes out of a predefined set, commonly referred in the literature as *line pool*. Each member of this set already defines the route and the stopping stations. However, in the model presented in this study, all possible combinations of stopping stations for each member of the route pool are considered.

2.3. **Safety and Operational Constraints**

A solution is feasible if it satisfies the following constraints:

Block seizing - in order to prevent a collision, only one train at a time is allowed at each block.

Block headway (H_b) - for trains traveling in opposite directions on block b, some extra time is required between the exit time of one train from the block and the entrance time of the next, so that the train has enough time to safely switch blocks.

Station headway (H_s) – defines the extra occupancy time of stations by trains that dwell there. This time is added to the actual time the train is scheduled to dwell at the station in order to enhance the safety and robustness of the timetable.

Block utilization (U_b) - defines the fraction of time that block b can be seized in a cycle. The utilization restriction enhances the robustness of the planned timetable.

Station capacity (V_s) – defines the maximum number of trains that can dwell at station s at the same time.

Minimal dwelling time (W_s) – minimal stopping time at station s required to allow embarking and disembarking of passengers.

Minimal transfer time (R_s) - the minimal time required for a passenger to switch trains at station s.

2.4. **Objective Function**

The objective function consists of two components, namely, the operational cost and the user inconvenience. The operational cost is proportional to total engine time and the user inconvenience cost is proportional to the passengers' total travel time. The *engine time* of a train is the time between the departure time from its origin station and the arrival time at its final destination. The *journey time* of a passenger is defined as the time between arriving at the origin and arriving at the destination assuming that the passenger selects the itinerary that minimizes this time.

The goal is to minimize a weighted sum of the total engine time and the total journey time. Given a timetable, it is easy to compute the total engine time of the trains. However, in order to determine the total journey time, each passenger's flow, $O_{s_1,s_2,t}$, has to be assigned to its optimal itinerary. Note that a feasible itinerary must comply with the scheduled trains and with the minimal transfer time. Let us denote the journey time of a passenger who arrives in station s_1 at time t and wishes to travel to station s_2 by $F(s_1, s_2, t)$. The total journey time is given by

$$\sum_{t=0}^{T} \sum_{s_1,s_2} O_{s_1,s_2,t} F(s_1, s_2, t) \quad (1)$$

An alternative approach is to formulate the model with bi-criteria objective namely the total engine time and the total journey time. The decision maker can then be provided with the entire efficiency frontier.

2.5. **Modeling Assumption**

Further assumptions in this model are as follows:

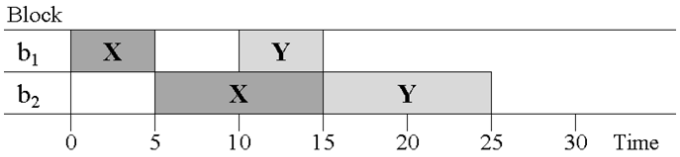
1. The total demand for journeys is unaffected by the offered service, i.e., the line plan and timetable. However, the actual demand for each train is affected by the individual itinerary decisions.
2. The arrival times of the passengers at their origin station is unaffected by the actual timetable. Passengers are assumed to arrive at the station continuously throughout the day, at inhomogeneous rates. This assumption is reasonable for regional systems where the frequency of trains is relatively high.
3. The capacity of the trains is not binding. This assumption is very common in the timetabling literature. In Section 6, we discuss how this assumption can be relaxed.
4. All the trains are identical in their maximal speed, although the traveling speed may vary from block to block.

3. Properties of Optimal Solutions

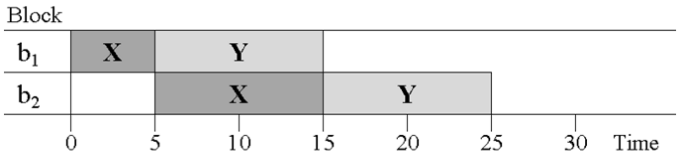
In this section, it is proven that under some reasonable conditions, trains should always be scheduled to travel at their maximal speed. This property is useful in the design of the insertion algorithm presented in the next section.

A set of several consecutive blocks that are not separated by stations are referred to as a *sequence of blocks*. Consider a sequence of two blocks, b_1 *and* b_2, that can be traversed by a train in five and ten minutes, respectively. Suppose that train X traverses the sequence at its maximal speed, starting at t=0. Consider a train Y that wishes to enter b_1 at t=5 minutes. Either the train should be delayed at a station before block b_1 for an additional five minutes (see Figure 1.a) or its speed on b_1

should be reduced (see Figure 1.b). A combination of slowing down the train on b_1 and delaying it before b_1 may also be used.



(a) Delaying train Y before block b_1



(b) Slowing down train Y on block b_1

Figure 1: Synchronization of two trains on a sequence of blocks

The above example demonstrates one aspect of the diversity of the set of feasible timetables. The goal here is to reduce the set of timetables that are being considered without losing too much in terms of optimality. Consider the two following mild assumptions:

Assumption 1: The capacity of the stations is not binding.

Assumption 2: All the railway intersections and interchanges occur at stations (either passenger or operational stations).

These assumptions may be violated in some railway systems, but such violations are rare and affect only a few trains. In particular, Assumption 2 is always valid for scheduling trains in a corridor. Under these two assumptions, it is shown that there exists an optimal solution for the SOLPTP where all the trains run at their maximum speed. To prove this claim, the following lemma is first proved.

Lemma 1: Under the assumptions above, any feasible timetable can be modified to a new feasible timetable such that

1. All trains run at their maximal speeds.
2. The entrance time to each sequence of blocks in the modified timetable is not earlier than the entrance time in the original timetable, and the exit time is no later.
3. The set of scheduled trains remains unchanged.

Proof: First note that by Assumption 2, trains traveling in opposite directions cannot simultaneously seize blocks in a sequence. In addition, trains that travel in the same direction cannot overtake each other within the sequence. Consequently, each block in the sequence is traversed by the same set of trains and in the same order.

In the original timetable, the entrance time of train i to the k^{th} block in the sequence is denoted by $t_{i,k}$, and the traveling time on it is denoted by $p_{i,k}$. Recall that in a feasible timetable, $t_{i,k} + p_{i,k} = t_{i,k+1}$. By convention, in a sequence of n blocks, $t_{i,n+1}$ is the exit time from the last block in the sequence.

Let τ_k denote the minimal traveling time on block k, assuming that the trains travel at their maximum speed. Let k^* be the block that maximizes τ_k, i.e., k^* is the longest block in the sequence.

A new timetable with entrance times denoted by $t'_{i,k}$ is constructed as follows

$$\boldsymbol{t'_{i,k^*+1} = t_{i,k^*+1}} \tag{2}$$

$$\boldsymbol{t'_{i,k} = t_{i,k^*+1} - \sum_{l=k}^{k^*} \tau_l \quad \forall k = 1, \dots, k^*} \tag{3}$$

$$\boldsymbol{t'_{i,k} = t_{i,k^*+1} - \sum_{l=k^*+1}^{k-1} \tau_l \quad \forall k = k^* + 2, \dots, n+1} \tag{4}$$

The entrance time to each block $k \leq k^*$ is shifted forward and the entrance time to each block $k > k^* + 1$ is shifted backward such that the train travels at the maximum speed and the entrance time to block $k^* + 1$ is left unchanged. In this new timetable, each block k is traversed in τ_k time units.

In the original timetable, the time difference between any pair of consecutive trains on block k^* satisfies $t_{i+1,k^*} - t_{i,k^*} \geq p_{i,k^*} \geq \tau_{k^*}$. Therefore, in the new timetable, the time difference between any consecutive trains on any block is at least τ_{k^*}, i.e., there are no conflicts between pairs of consecutive trains.

Next, note that by Assumption 1, changing the departure and arrival times at the stations cannot cause any conflict at a station. Finally, because $p_{i,k} \geq \tau_k \quad \forall i, k$ it follows that $t'_{i,k} \geq t_{i,k} \quad k = 1, \dots, k^*$and $t'_{i,k} \leq t_{i,k} \quad k = k^* + 1, \dots, n+1$, and in particular, $t'_{i,1} \geq t_{i,1}$ and $t'_{i,n+1} \leq t_{i,n+1}$. As a consequence, the entrance time of each train to the sequence in the new timetable is not earlier than its entrance in the

original timetable, and the exit time of each train from the sequence in the new timetable is not later. **Q.E.D**

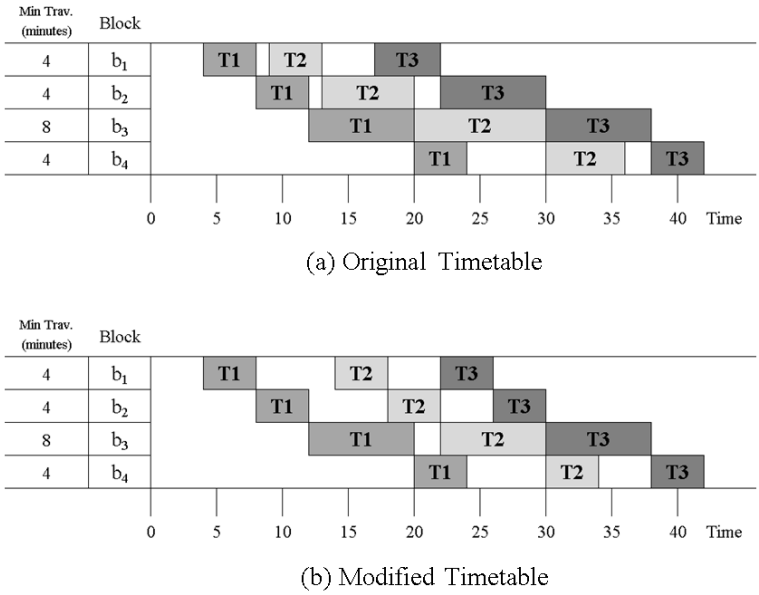


Figure 2: Rescheduling trains in a sequence of blocks to facilitate travel at the maximal speed

In Figure 2, the rescheduling procedure used by the proof of Lemma 1 is demonstrated. The example consists of a sequence of four blocks (b_1, b_2, b_3, b_4) and three trains ($T1, T2, T3$). The longest block is b_3, with a minimal traversal time of eight minutes. An arbitrary feasible schedule is presented in Figure 2(a). In Figure 2(b), we present a new schedule for the same trains that is constructed as follow: The traversal times of all trains on all blocks are changed to the minimal traversal times. The exit times from b_3 are kept unchanged, while the entrance times of $b_1, b_2,$ and b_3 are shifted forward and the exit times from b_4 are shifted backward. While the travel times on the sequence of the three trains in the original timetable (a) are 20, 27 and 25 minutes, respectively, the travel time in the revised timetable (b) is 20 minutes for all trains.

Note that the claim of Lemma 1 is similar to the claim that in a no-wait flow-shop model with identical jobs, solutions where the prolonging of operations is not used are dominant. See for example Abadi et al. (2000). Next, this observation is used to prove that under the assumptions above, traveling at the maximum speed is dominant with respect to the objective of minimizing the total journey time.

Proposition 1: There exists an optimal solution, with respect to the objective functions of minimizing total journey time of all passengers and total engine time, where all trains travel at their maximum speed on all blocks.

***Proof*:** By Lemma 1, it is always possible to modify any solution such that all blocks are traversed at the maximum speed by moving the departure time from all stations forward and moving the arrival time at all stations backward. Consider two stations A and B connected by a sequence of blocks that is traversed by a scheduled train. Denote the original (resp., modified) departure time from A by t_1 (resp., t_1') and the arrival time at B by t_2 (resp., t_2').

Consider the effect of the timetable modification on three groups of passengers:

1. Passengers that boarded the train at A
2. Passengers that disembarked the train at B
3. All other passenger in the system

Some members of the first group are better off. Because the departure time from A is postponed, some passengers have the opportunity to catch earlier trains. Members of the second group are clearly better off because their journey time is shorter. Finally, all other passengers, including those that are on board trains that travel from A to B are indifferent.

The total engine time of all trains can only be reduced. This is because the departure time from the first station and the arrival time to the last station may be shifted forward and backward, respectively.

Q.E.D

4. Algorithm

The complexity of the SOLPTP calls for a heuristic approach. The solution method presented in this paper is based on the paradigm of the cross-entropy (CE) metaheuristic, introduced by Rubinstein (1999). CE is an evolutionary metaheuristic that iteratively applies the following two phases:

1. Generation and evaluation of a sample of random solutions according to a specified random mechanism.
2. Updating the parameters of the random mechanism on the basis of these solutions in order to produce a "better" sample in the next iteration.

For additional references on CE for combinatorial optimization problems, see Rubinstein and Kroese (2004). CE has been applied to various problems, e.g., vehicle routing problems, Chepuri and Homem-de Mello (2005), and stochastic project scheduling problems, Bendavid and Golany (2009).

In Figure 3, an outline of the CE heuristic for SOLPTP is presented. A set of coded solutions are generated by an initial arbitrary random mechanism. These solutions are decoded into line planes and cyclic timetables and are then evaluated. The random mechanism is updated based on the values of these solutions, and the process is repeated until some stopping criteria are met.

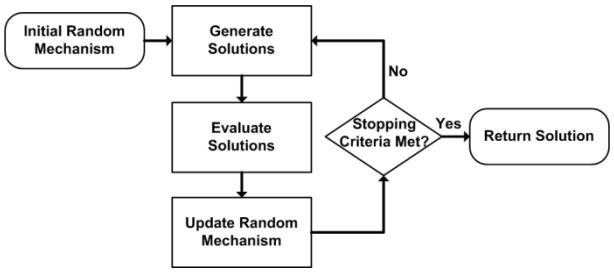


Figure 3: Outline of the CE algorithm

The rest of the section is organized as follows: a method for encoding solutions is presented in subsection 4.1. In subsection 4.2, procedures to construct and evaluate solutions are introduced. The updating procedure of the random mechanism is described in subsection 4.3.

4.1. **Solution Encoding**

A solution is encoded by a set of *genes* referred to as a *chromosome*. Each gene represents a possible train and corresponds to a route in the route pool. The number of genes related to any given route is predetermined by the planner and is bounded by the maximal frequency of the route. A *gene* contains the following information:

- In Use (IU) – A Boolean that indicates whether or not this train is being used in this solution.
- Gate Time (GT) – An integer that indicates the earliest time (in minutes) in which the train can enter its first block.
- Stopping Stations (SS) – Boolean vector with an element for each station along the train's route. A value of 1 in the j^{th} element of this vector indicates that the

train stops at its j^{th} station. Otherwise, the train may pass the j^{th} station without stopping there.

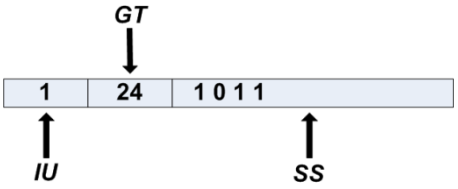


Figure 4: A gene

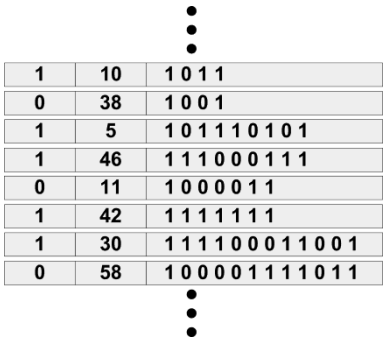


Figure 5: A part of a chromosome

Figure 4 and Figure 5 demonstrate the structures of a gene and a set of genes that represent a part of a chromosome. The structure of all the chromosomes of a given system is identical. The relationship between genes and routes is stored separately and once for all the chromosomes.

4.2. **Solution Evaluation**

The evaluation of a chromosome consists of four stages. First, a feasible solution is constructed, i.e., a line plan and a cyclic timetable. Second, a graph representation of all feasible passenger itineraries is constructed. Third, the shortest path from each node in this graph to each station is calculated in order to obtain the optimal itineraries for each flow of passengers. Finally, the total journey time of all passengers and the total engine time are calculated. The entire process is outlined in Figure 6. The stages of this process are described in the rest of this subsection, as depicted in the figure.

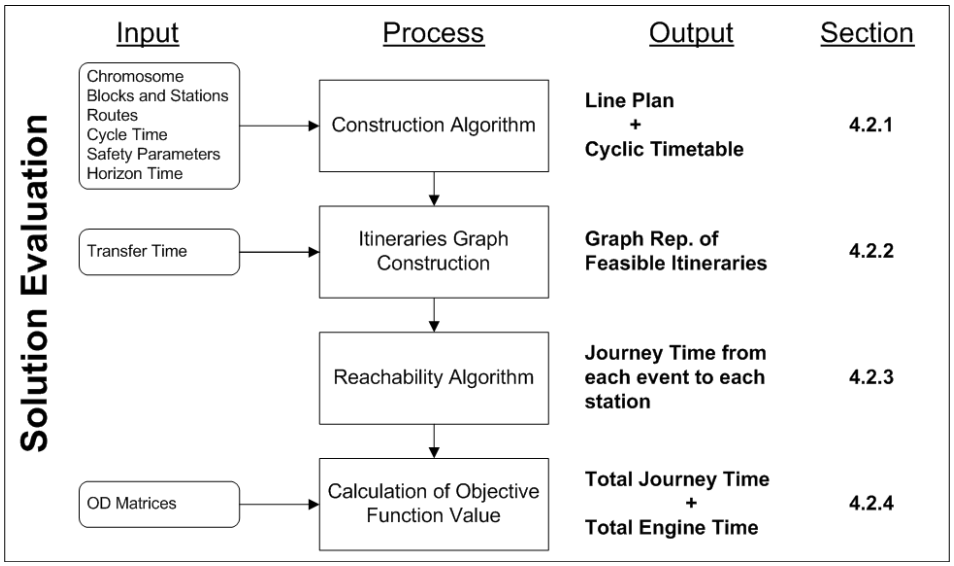


Figure 6: Outline of the solution evaluation process

4.2.1. Construction Algorithm

A solution is constructed based on a chromosome in two stages. First a line plan and a single feasible cycle of the timetable are constructed, and then the cycle is replicated over the entire planning horizon (e.g., a day).

A greedy insertion algorithm is used to construct a line plan and a feasible cycle of the timetable. Recall that the length of a cycle is denoted by C, and it is typically 60 minutes. The algorithm attempts to schedule possible trains with true "in use" indicators ($IU = 1$) one-by-one in an increasing priority order (ties are broken by instance number of the train within the route and then by route number). It is possible to incorporate the priority parameter as a decision variable into the CE search mechanism.

A train is inserted into the blocks and stations according to the order prescribed by its route. The gate time for the first block is retrieved from a gene (GT value). The gate time for the next block is the exit time from the previous ones, plus possibly, the minimal dwelling time (W_s) at the station that follows it. A train is scheduled only if it can be inserted into all its blocks and stations.

For each block, the algorithm keeps track of the time slots within a cycle that have been seized by previously scheduled trains. In order to insert a train into a block, the algorithm searches for the earliest interval that has a length no shorter than the minimum traversal time of the block that is not seized. If other trains are scheduled to

run in the opposite direction on the block, a headway time of H_b is enforced. The train cannot be inserted if such an interval does not exist or if the insertion causes a violation of the utilization constraint (U_b).

For a sequence of blocks, the exit and entrance times of each pair of consecutive blocks have to be equal. If the entering time to a block is later than the exit time from the preceding block, the algorithm tries to extend the seizing time on the previous block. If this extension is not possible, the train cannot be inserted into the sequence of blocks. If an insertion to the sequence is possible, the algorithm modifies the seizing times, as explained in the proof of Lemma 1, so that the sequence is traversed in minimal time.

The algorithm keeps track of the occupancy of each station at any period in the cycle. Recall that the cycle is divided into short periods, typically of one minute each. A station is seized by a train starting at its entrance time and until H_s periods after its departure to the next block. The algorithm does not choose the platforms for the trains in each station. However, the station capacity is respected (V_s). Hence, the resulted platforming problem is likely to be solvable.

The output of the insertion algorithm is a line plan and a feasible cycle. Note that the route frequencies are implied by the number of scheduled trains per route. In addition, the insertion algorithm determines the stopping stations of each train. Both of these are elements of the line planning problem. A formal description of the insertion algorithm is given in Table 1.

Replication of a cycle over the planning horizon is straightforward. We note, however, that because the traveling time of a train may expand over several cycle periods, incomplete trains should be removed at the end of the planning horizon.

Table 1: Insertion Algorithm – constructing a cycle of the timetable based on a chromosome

```
Create a list of all genes having IU = 1 sorted by their route priorities
For all genes g in the list
    Set t to g.GT;
    Let block_list be the list of all blocks of the route of g sorted by the travel direction of the route
    For each block b in the block_list
        Find the earliest interval of time periods in the cycle that start not earlier than t in which the g can be inserted
        If unable to insert, then
            break;   (this train cannot be scheduled)
        If current block follows a station, then
            Find the time periods in which the train dwells at the station;
            Increase occupancy of the station at these periods and the following H_s periods;
            If Station Capacity is violated, then
                Break;   (this train cannot be scheduled)
        Else
            If exit time from previous block is not equal to enter time to current block
                If it is possible to extend train's seizing of the previous block, then
                        Extend the train's seizing of the previous block;
                Else,
                        Break; (this train cannot be scheduled)
            If this block is followed by a station, then
                Try to shorten and shift all previous block seizing forward in the sequence;
        Set t to the exit time from current block
        If this block is followed by a station j with SS_j = 1, then
            Set Time to (t+ W_s) modulo C
    If Station Capacity is violated, then (handle last station of the route)
        break;   (this train cannot be scheduled)
```

4.2.2. Itineraries Graph Construction

In order to evaluate the total journey time of all passengers given a timetable, we create a graphical representation of all possible itineraries. The *itineraries graph* is a directed graph consisting of a node for each event in the timetable. A node is characterized by station, train, type (arrival/departure), and time.

Each arrival node is connected by an arc to the following nodes:

- The next arrival node of the same train (in the next station).
- The earliest departure node from the same station at a time that is at least R_s later than the node's time

Each departure node is connected by an arc to the following nodes:

- The next arrival node of the same train (in the next station)
- The next departure node in the same station

The length of each arc is the time difference between its end nodes. Each path on the itineraries graph represents a feasible itinerary, and the length of the path represents its travel duration, including transfer times. The structure of this graph allows itineraries with transfers subject to the minimal transfer time limitations at the various stations.

A sample timetable and its corresponding itineraries graph are given in Figure 7. The timetable (Figure 7.a) consists of tree trains, train T15 traveling from station A to station C and trains T17 and T21 traveling from station C to station D. In the corresponding itineraries graph (Figure 7.b), arrival and departure nodes are colored in grey and black, respectively. A passenger that enters the system at station C before 09:11 will be able to board train T17 or wait for a following train at station C. Hence, the departure node of train T17 at station C is connected to the arrival node at station D and to the following departure node at station C. A passenger that arrives on train T15 at station C and wishes to continue to station D will not be able to board train T17 because of the transfer time limitation. Hence, the arrival node of train T15 to station C is connected to the departure node of train T21, the first departure node that is at least the *minimal transfer time* later.

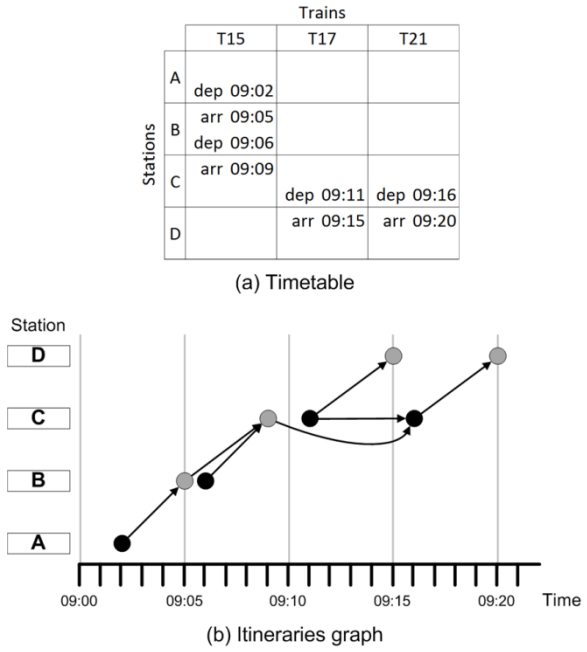


Figure 7: Sample timetable and a corresponding itineraries graph

Müller-Hannemann et al. (2007) introduce a similar but more general graphic representation of itineraries with transfers referred to as the *realistic time-expanded*

graph. However, their representation requires approximately twice the number of nodes and arcs. Note that because the *itineraries graph* is created numerous times in the course of the CE algorithm, a more compact representation is beneficial.

4.2.3. Reachability Algorithm

To evaluate a solution, the journey time function $F(s_1, s_2, t)$ of each demand flow needs to be calculated. The structure of the itineraries graph assures that any path of the graph is a feasible itinerary. Calculating $F(s_1, s_2, t)$ is equivalent to finding the shortest path from the departure node in the origin station following the passenger's arrival to a node in the destination station. There are known algorithms for all-pairs shortest paths, such as the Floyd Warshall algorithm with complexity of $O(N^3)$ and Johnson's algorithm with complexity of $O(N^2 \log N + NA)$, where N is the number of nodes and A is number of arcs. See for example Cormen et al. (2001).

However, two special properties of the itineraries graph can be exploited in order to devise a more efficient algorithm for finding the optimal itineraries of all passengers. First, the graph has a special structure; it is a directed acyclic with a maximal out degree of two. Second, although for each departure node, there might be several paths to arrival nodes at each station, only the earliest arrival node is of interest.

The *reachability time* is referred to as the earliest time that a station can be reached from a given node. A vector that stores the reachability times to all stations is defined for each node. The reachability algorithm is based on the fact that the reachability time from a node is the minimum between the reachability times of the two nodes that it is connected to. In order to calculate the reachability times, the algorithm advances through the nodes in a nonincreasing order of their time. In Table 2 below, a dynamic programming formulation of the reachability algorithm is presented.

The reachability algorithm has two main steps:

- Sorting the nodes of the itineraries graph in a nonincreasing order of time. The complexity of this operation is $O(N \log N)$, where N is the number of nodes.
- Calculating the reachability times for each node. The complexity of this phase is $O(NK)$, where K is the number of passenger stations in the system.

Typically, $log\ N \ll K$, so the complexity of the algorithm is determined by the second step. In this case, the complexity of the algorithm is $O(NK)$, which is linear in the output size. In the unlikely situation where $log\ N > K$, the complexity of the algorithm is $O(N \log N)$. In an ad hoc implementation of the algorithm, instances with some 10,000 events (nodes) and 47 stations could be solved in less than 0.07 seconds on a modest desktop computer.

Table 2: Dynamic programming formulation of the reachability algorithm

$ET(n,s)$ - Earliest time from node *n* to station *s*
t_n - Node's time
S_n - Node's station
σ_n- Following node in station S_n
ρ_n - Following node of the same train
$ET(n,s) = \min\{ET(\sigma_n,s), ET(\rho_n,s)\}$
Proceed in nonincreasing t_n
If $S_n = s$, then $ET(n,s) = t_n$
If σ_n is undefined, then $ET(\sigma_n,s) = \infty$
If ρ_n is undefined, then $ET(\rho_n,s) = \infty$

While the itineraries graph and reachability algorithm were devised for the optimization algorithm introduced in this paper, they can also serve as components of a decision support system (DSS) in which a timetable is constructed manually. In addition, these are a more efficient alternative to solving the earliest arrival problem presented by Müller-Hannemann et al. (2007) in the context of supporting passengers' personal itinerary decisions.

4.2.4. Calculation of the objective function value

The values of the two components of the objective function are calculated in the following manner: The engine time of a train is the time difference between its latest and earliest events. We obtain the total engine time by summing over all the trains in the timetable. To calculate the journey times of all passengers, the journey time of each passenger's flow is calculated. Given the reachability times from each node of the itineraries graph, the journey time $F(s_1, s_2, t)$ of flow $O_{s_1,s_2,t}$ is calculated by the following equation:

$$\boldsymbol{F(s_1, s_2, t) = ET(d, s_2) - t} \quad (5)$$

where d is the first departure node at station s_1 following the passenger's arrival. The journey times of all the passengers is calculated by Equation (1).

The objective function is a weighted sum of the total engine time and the total journey time. The weights reflect the ratio between the cost of an engine hour and the cost of a "passenger hour" as viewed by the planner.

4.3. **Updating the Random Mechanism**

Recall that each solution is represented by a chromosome that is built of a set of genes. Each gene consists of Boolean variables (IU and SS) and an integer variable (GT). For each of these variables, a probability function is created. The Boolean variables are sampled from a Bernoulli distribution, and the integer variables are sampled from a general discrete probability function.

The initial parameters of the Bernoulli distributions are set to 0.5. The distributions of GT are initially set to be Uniform over the set $\{0, \dots, C-1\}$. These choices are arbitrary but do have a minor effect on the value to which the CE algorithm converges.

Based on these distributions, a generation of chromosomes is created. The number of such chromosomes is denoted by *GS* (for generation size). After constructing feasible solutions based on a generation and evaluating them, an *elite group* is selected, which is the γ lower (best) quantile of the solutions. The updating mechanism of the distributions to be used in the following iteration of the algorithm is defined next. For this purpose, the following notation is introduced:

$\mu_{i,t}$ distribution parameter of IU of gene *i* at iteration *t.*

$v_{i,j,t}$ the distribution parameter of SS of the j^{th} 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 i^{th} gene of solution *k* at iteration *t*

$SS_{k,i,j,t}$ SS value of the j^{th} stop of the i^{th} gene of solution *k* at iteration *t.*

$GT_{k,i,t}$ GT value of the i^{th} gene of solution *k* at iteration *t*

The distribution parameters are updated according to the following equations:

$$\mu_{i,t} = \alpha \cdot \frac{\sum_{k \in elite} IU_{k,i,t}}{\gamma \cdot GS} + (1-\alpha) \cdot \mu_{i,t-1} \quad (6)$$

$$v_{i,j,t} = \alpha \cdot \frac{\sum_{k \in elite, IU_{k,i,t}=1} SS_{k,i,j,t}}{\sum_{k \in elite} IU_{k,i,t}} + (1-\alpha) \cdot v_{i,j,t-1} \quad (7)$$

$$\eta_{i,r,t} = \alpha \cdot \frac{\left|k: k \in elite, IU_{k,i,t} = 1, GT_{k,i,t} = r\right|}{\sum_{k \in elite} IU_{k,i,t}} + (1-\alpha) \cdot \eta_{i,r,t-1} \quad (8)$$

where α is a smoothing parameter that helps to prevent premature convergence of the algorithm. If the denominator of Equations (7) and (8) equals zero, then these parameters are left unchanged, that is, $v_{i,j,t} = v_{i,j,t-1}$ and $\eta_{i,r,t} = \eta_{i,r,t-1}$.

4.4. **Stopping Criteria**

The random mechanism is repeatedly employed to create generations of solutions, and it is updated after each generation, as described above. The algorithm is stopped when one of the following criteria is met:

Convergence of the distribution parameters - stop when both of the conditions below hold:

1. For all genes (trains) i, one of the following is true: (a) $\mu_{i,t} < \epsilon$, (b) $\mu_{i,t} > 1 - \epsilon$, or (c) the train was not scheduled at all in the solutions of the last generation.
2. For all genes (trains) i having $\mu_{i,t} > 1 - \epsilon$, the values of all the parameters $v_{i,j,t}$ and $\eta_{i,r,t}$ are either greater than $1 - \epsilon$ or smaller than ϵ.

where ϵ is a small positive number, say $\epsilon = 0.01$.

Convergence of the objective function value - stop when there is no improvement in the objective function value during the last ζ iterations, where ζ is a parameter of the algorithm.

Time limit - stop when a predefined time limit or number of iterations is exceeded.

5. Numerical Experiments

In this section, we present the results of an extensive numerical experiment that was conducted to verify the effectiveness of the proposed CE algorithm and to calibrate its parameters. Because there are no previous studies on the service-oriented line planning and timetabling problem, our results are compared with results obtained by a team of railway planners at Israel Railways. This team worked on the same instance and had the same goal of minimizing user inconvenience and operational costs.

Israel Railways is an independent government-owned corporation. The company provides passenger and freight transportation and is responsible both for the infrastructure and the rolling stock. Because the entire system is managed by a single organization, a centralistic planning approach can be applied. The infrastructure consists of approximately 1000 kilometers of rail tracks, 47 passenger stations and 30 operational and freight stations. The policy of Israel Railways is to give a clear priority to passenger trains. The scheduling of the freight trains is done subsequently to the scheduling of the passenger trains, and the utilization of the infrastructure by passenger trains is taken as a hard constraint for the freight scheduling problem. Therefore, the passenger timetabling problem is solved as if the infrastructure is only used for passenger transportation. For more information about Israel Railways, see http://www.rail.co.il/En/.

The complete data for this instance can be downloaded from http://www.eng.tau.ac.il/~talraviv under "Publications". The web page contains an archive file with several data tables and a detailed description of their structure. In Appendix A, a graphical description of the infrastructure is presented. The figure shows the logical relations between the different blocks and stations.

In the experiments presented below, the algorithm was tested with the 28 routes used in the timetable of 2008. For the sake of fair comparison, the possible benefit from additional routes was not checked, although such routes could be easily added. The priority of the routes was set such that intercity trains were given precedence over metropolitan ones. The maximal frequency of all the routes was set to four. The safety and operational parameters were set to the same values used by Israel Railways planners, as given in Table 3.

Table 3: safety and operational parameters of the case study instance

Block headway (H_b)	1 min.
Station headway (H_s)	1 min.
Block utilization (U_b)	75%
Minimal dwelling time (W_s)	1 min.
Minimal transfer time (R_s)	4 min.

The weight of the total journey time was set to one, and the weight of the total engine time was varied across three different values: 0, 100, and 200. The first option ignores operational costs completely and concentrates on reducing total journey time.

With a weight of 100, the value of each engine hour equals 100 passenger hours. Note that while the planners wish to minimize both user inconvenience and operational cost, they typically cannot specify the exact ratio between the weights.

Observe that the effect of the weight ratio on the nature of the optimal solution is very significant. When the operational cost is high, the optimal solution is likely to include fewer trains that are in turn easier to schedule given the same limitations. Hence, using different weight ratios results in considerably different problem instances.

5.1. **Tuning Experiment**

In order to calibrate the CE algorithm parameters and to check the sensitivity of the algorithm to its parameters, a full factorial experiment was conducted with the following three algorithm parameters:

1. Smoothing - with two levels $\alpha = 0.3$ and $\alpha = 0.7$.
2. Generation size - with two levels *GS*=500 and *GS*=1000. As recommended by previous studies that applied the CE method, the elite size is set to be 10% of the generation size.
3. Keep Elite (*KE*) –the algorithm was tested with and without keeping the elite group for the next generation, a concept known in the evolutionary algorithm literature as *elitism*.

In the insertion process, some genes having $IU = 1$ may eventually not be scheduled. Another parameter that was considered was whether to update the distribution parameters according to genes that have $IU = 1$ or only according to genes that were actually scheduled during the insertion process. A few preliminary runs showed a great difference between the two in favor of the first option. Therefore, this parameter was omitted from the experiment. It seems that ignoring trains that could not be scheduled may prematurely disqualify too many trains out of the generated solutions.

The CE algorithm was implemented in C and was tested on an Intel® Xeon™ 2 CPU E5506, 2.13 GHz. In order to save time, the four cores of the CPU were exploited by launching two runs in parallel. A full $3 \cdot 2^3$ factorial experiment with four replications was carried out in order to test the effects of the three algorithm parameters on the three instances of the problem. Six hours were allocated for each

run; in order to examine the convergence of the algorithm, the stopping criteria were not activated. Summarized results of the 96 runs are given in Table 4: the settings of the runs are given in the first four columns. The fifth and sixth columns present the average and range of the four runs of each setting. The minimal value obtained for each setting is given in the seventh column, the average number of trains scheduled per hour is given in the eighth column and the average number of generations created in six hours is given in the last column. An ANOVA analysis showed that the main factors, α and *GS*, had a significant but rather small effect in terms of the objective function value. In addition, for each problem instance, the gap between the average solution of the best and worse settings was less than 1.7%. This may suggest that the algorithm is not extremely sensitive to changes in its parameters, which is good news.

It is apparent from the results of the experiment that no set of algorithm parameters dominates the others for all problem instances. For $ratio = 0$, the best settings are $\alpha = 0.7, GS = 1000, KE = Yes$, and for $ratio = 100$ and 200, the best settings are $\alpha = 0.3, GS = 1000, KE = No$. However, both of these settings performed well in all three instances.

Table 4: Experiment results

Run Setting				Objective Function Value (Hours/Day)			Trains Per Cycle	Number of Generations
Ratio	α	**GS**	**KE**	**Average**	**Range**	**Minimum**	**Average**	**Average**
0	0.3	500	No	100,101.5	858.7	99,689.0	41.25	514.75
0	0.3	500	Yes	100,384.2	703.0	100,122.4	40.00	605.25
0	0.3	1000	No	100,033.9	1,179.0	99,292.5	38.50	285.25
0	0.3	1000	Yes	101,080.2	2,254.0	100,153.0	37.00	331.50
0	0.7	500	No	101,330.6	1,532.7	100,796.6	42.25	509.25
0	0.7	500	Yes	100,785.3	1,444.1	100,177.8	42.00	554.00
0	0.7	1000	No	100,425.6	1,803.0	99,418.3	38.75	256.25
0	0.7	1000	Yes	99,643.2	991.8	99,125.6	40.25	320.75
100	0.3	500	No	155,178.2	1,195.1	154,620.7	16.25	1,019.50
100	0.3	500	Yes	155,009.1	990.6	154,277.9	14.75	1,171.75
100	0.3	1000	No	154,182.3	659.4	153,755.0	14.75	494.50
100	0.3	1000	Yes	155,134.7	2,063.1	154,439.9	15.25	561.75
100	0.7	500	No	156,326.7	2,768.3	154,963.6	18.00	1,024.50
100	0.7	500	Yes	156,389.5	1,726.8	155,842.6	17.75	1,155.75
100	0.7	1000	No	155,082.4	1,221.2	154,471.9	15.00	503.75
100	0.7	1000	Yes	154,737.8	1,141.5	154,214.4	15.25	562.75
200	0.3	500	No	186,228.4	787.0	185,765.1	11.00	1,253.00
200	0.3	500	Yes	186,804.3	1,999.5	186,000.6	10.50	1,371.00
200	0.3	1000	No	185,562.6	1,501.7	184,930.3	10.25	595.25
200	0.3	1000	Yes	185,563.5	1,026.5	185,091.1	10.25	664.00
200	0.7	500	No	187,312.4	1,242.5	186,776.4	12.75	1,250.50
200	0.7	500	Yes	187,980.8	2,502.5	186,825.5	12.75	1,395.50
200	0.7	1000	No	185,890.2	1,006.6	185,466.1	10.50	615.50
200	0.7	1000	Yes	185,859.6	498.1	185,562.5	10.50	672.25

The *Ratio* factor had a significant, positive, large effect. Clearly, the system is better off with smaller operational costs when the user inconvenience is kept constant. Interestingly, the algorithm produces a greater number of solutions (and hence generations) with larger operational costs within the allotted time. This is because the random mechanism adjusts itself to produce solutions with fewer trains when the costs are higher. The evaluation of such solutions is carried out more quickly.

5.2. **Running Time and Convergence of the Algorithm**

In this subsection, some observations on the convergence process of the algorithm are made, and conclusions regarding the stopping criteria are derived. For all the runs, the best solution and completion time of each generation was recorded. Figure 8 presents the evolution over time of the average solution values of the three instances. In the first half hour, a sharp decrease in the solution value is noticeable for all three instances. The solutions of the $ratio = 200$ and $ratio = 100$ instances did not further improve after approximately one and a half hours and two hours, respectively. However, for the $ratio = 0$ instance, the graph is still decreasing after six hours. Hence, it seems that the solutions could have improved slightly if more time was allocated for this instance.

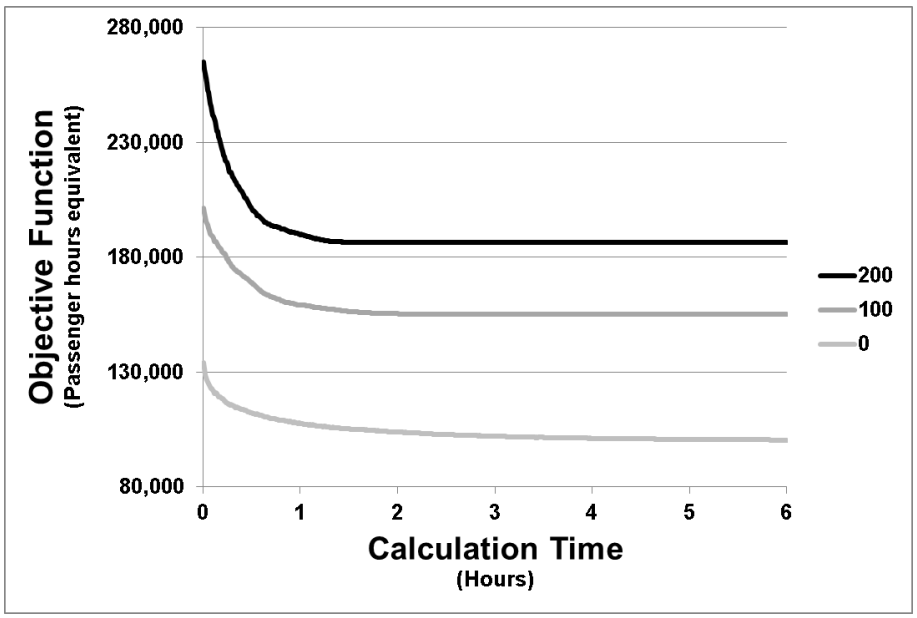


Figure 8: Convergence of the average solution per problem instance (ratio)

To further examine the convergence process in each setting, additional information is presented in Table 5. The first four columns of the table display the run settings. The first time at which the best solution of the run was obtained is presented in the fifth column. The first time at which a solution within 1% of the best one was encountered is presented in the sixth column. In addition, three values of the *no-improvement* stopping criterion parameter were inspected, namely, $\zeta = 15, 30, 50$ generations. The seventh column of Table 5 presents the average time in which the $\zeta = 15$ stopping criterion was triggered. The eighth column presents the average difference between the solution when the criterion was triggered and the best solution obtained in the same run. An (-) indicates that the corresponding stopping criterion was not triggered. In the rest of the table, the same information is presented for $\zeta = 30, 50$.

It is apparent from the table that while the best solutions of the $ratio = 100$ and $ratio = 200$ instances are found long before the time limit (21600 seconds) expires, the best solution of the $ratio = 0$ instance is found toward the end of each run. Indeed, as we already observed, the six hours limit is quite sufficient for the former instances but not for the later.

Table 5: Testing the Stopping Criteria

Run Setting				Initially Encountered (seconds)		No-improvement Stopping Criterion					
						15 Generations		30 Generations		50 Generations	
Ratio	α	**GS**	**KE**	**Best solution**	**1% from best**	**% from best**	**Time (seconds)**	**% from best**	**Time (seconds)**	**% from best**	**Time (seconds)**
0	0.3	500	No	19,594	8,124	8.0963%	5,003	-	-	-	-
0	0.3	500	Yes	21,331	9,513	14.3531%	1,569	-	-	-	-
0	0.3	1000	No	21,639	17,288	-	-	-	-	-	-
0	0.3	1000	Yes	21,531	18,323	11.5581%	3,870	-	-	-	-
0	0.7	500	No	18,098	2,974	0.0920%	8,137	0.0117%	13,281	0.0046%	14,629
0	0.7	500	Yes	17,482	4,369	4.2848%	5,543	0.0357%	14,405	0.0292%	17,095
0	0.7	1000	No	19,267	7,204	2.2827%	12,826	-	-	-	-
0	0.7	1000	Yes	21,116	14,627	10.0822%	4,504	-	-	-	-
100	0.3	500	No	8,764	3,137	0.1339%	4,703	0.0365%	6,106	0.0089%	7,586
100	0.3	500	Yes	7,873	2,736	4.5859%	2,935	0.0957%	5,653	0.0652%	6,685
100	0.3	1000	No	10,686	6,673	7.4005%	6,208	0.0045%	11,411	0.0000%	12,785
100	0.3	1000	Yes	13,086	6,182	8.9785%	4,877	0.0750%	10,868	0.0000%	14,954
100	0.7	500	No	4,567	1,227	0.1732%	2,338	0.0351%	3,616	0.0131%	4,647
100	0.7	500	Yes	7,423	1,468	0.1863%	3,053	0.0463%	4,987	0.0463%	5,359
100	0.7	1000	No	8,282	2,845	0.2240%	5,236	0.0206%	7,910	0.0022%	9,766
100	0.7	1000	Yes	8,749	3,006	0.0273%	5,688	0.0230%	6,564	0.0099%	7,956
200	0.3	500	No	5,428	2,505	5.8040%	2,759	0.0880%	4,402	0.0092%	5,728
200	0.3	500	Yes	5,667	2,219	0.1776%	3,465	0.1104%	4,197	0.0632%	5,227
200	0.3	1000	No	8,332	5,082	0.0068%	7,869	0.0068%	8,387	0.0054%	9,450
200	0.3	1000	Yes	5,980	4,418	0.0398%	6,166	0.0000%	6,919	0.0000%	7,545
200	0.7	500	No	4,149	999	0.2324%	1,647	0.0310%	3,105	0.0310%	3,449

200	0.7	500	Yes	4,815	1,017	0.1365%	1,784	0.1254%	2,113	0.0246%	3,393
200	0.7	1000	No	7,531	2,129	0.1538%	3,436	0.0207%	5,944	0.0068%	6,970
200	0.7	1000	Yes	8,572	2,399	0.0246%	4,311	0.0057%	5,166	0.0057%	5,796

Using the no-improvement criterion with $\zeta = 30{,}50$ could save a great deal of running time for the $ratio = 100, 200$ instances at negligible cost. This stopping criterion was not triggered in most of the $ratio = 0$ runs. Indeed, the objective function value in these runs did not converge within the six-hour limit. However whenever the stopping criterion was triggered, the solution obtained was equal or very close to the best solution of the run. This implies that the no-improvement stopping criterion is a reasonable criterion that can be used when there is no definite time budget. Note that the no-improvement criterion with $\zeta = 15$ was triggered prematurely in some of the runs, and thus, it is not recommended.

We also tested the convergence of the distribution parameter criterion with $\epsilon = 0.1, 0.05, 0.001$. This criterion was rarely triggered prior to the no-improvement criterion. However, in the few cases during which the criterion was triggered, the obtained solution was equal or very close to the best one obtained within the six-hour time limit. We recommend the use of the distribution parameter criterion in addition to the no-improvement criterion if it is desirable to reduce the average running time.

While the evaluation of each solution is done by an efficient reachability algorithm, as described in 4.2.3, it consumes the major share of the computation time. Luckily, this part of the algorithm lends itself easily for parallelization because the evaluation of each sampled solution can be carried out independently. Hence, it is possible to reduce the running time proportionally to the number of processors.

5.3. **Comparing CE with a Brute Force Approach**

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. In order to refute this argument, the following test was performed: the initial CE distributions were used to randomly generate as many feasible solutions as possible to evaluate within a time limit of six hours. This is in fact equivalent to running the CE algorithm with α=0. This test was replicated four times for each problem instance.

Table 6: Comparing CE with a Brute Force Approach

Ratio	Total Number of Solutions	Brute Force Best (Hours/Day)	Difference from average of all CE runs	Difference from average of best setting runs
0	1,851,000	122,225.4	21.65%	22.66%
100	1,846,000	187,995.8	21.09%	21.93%
200	1,840,000	242,782.8	30.25%	30.84%

In Table 6, we present a comparison of the result obtained by this approach to average results obtained by the CE algorithm. The first column identifies the instance by the cost ratio, the second column presents the total number of solutions in all four runs that could be produced and evaluated within the time limit, and the value of the best solution that was generated in all four runs is presented in the next column. The relative difference between the brute force solution and the average solution over all possible settings is presented in the fourth column. In the rightmost column, we present the relative difference compared with the average solution obtained by the best algorithmic setting of each instance. It is apparent from Table 6 that the results obtained by the CE are significantly better than those that could be obtained by a nonintelligent random approach.

5.4. **Comparison with the Current Timetable**

Next, the algorithm is benchmarked against the actual timetable being used by Israel Railways. This timetable was designed by a team of expert planners over a period of one year. It should be noted, however, that the actual planning task consists of solving other subproblems, such as platforming and rolling stock circulation, which are not treated here.

The timetable constructed by Israel Railways has a cyclic nature but is not entirely cyclic. The construction of this timetable was done by planning a cyclic timetable and later "removing" some trains at slack hours. In order to make a more accurate comparison, a fully cyclic timetable was constructed out of the published timetable. This was done simply by reinstating the removed trains. This timetable is referred to as the *peak timetable,* while the original one is referred to as the *published timetable*.

The total journey time of passengers and the total engine time of the published and the peak timetables were calculated as explained in 4.2.4. In order to compare the solutions obtained by the CE algorithm with the existing timetable, the ratio between operational costs and user inconvenience must be specified. Because this ratio is unknown, an efficiency frontier is constructed in order to visualize the trade-off between the two components. In order to plot an efficiency frontier, the CE algorithm was applied iteratively for various cost ratios, namely, 0, 25, 50, 75, 100, 150, 200, 400, and 800. The algorithm was employed with the following setting ($\alpha = 0.3, GS = 1000, KE = 0$), which yielded the best average among the ratio=0, 100, and 200 instances described in subsection 5.1. Each approximately optimal solution obtained is decomposed back to its passenger and engine components. This constitutes a point on the approximated efficiency frontier.

Figure 9 presents the approximated efficiency frontier constructed for the Israel Railways case. The horizontal axis represents the operational cost (in terms of engine hours), and the vertical axis represents the user inconvenience (in terms of total journey time). Each nondominated solution obtained by the algorithm is marked on the graph by a black circle. The published timetable is marked by a rhombus, and the peak one, by a square.

It is apparent from Figure 9 that it is possible to reduce the user inconvenience and the operational cost simultaneously. For example, the total journey time can be reduced by approximately 22% at the same operational cost. This represents a yearly saving of more than 7,650,000 passenger hours and shortens the average journey of a passenger from 65 to 51 minutes. In addition to the direct economic benefit obtained by saving so many working hours, there are clear indirect environmental and economic benefits in making the train a more attractive mode of transportation.

Because of the approximation error of the CE algorithm, Pareto-dominated solutions were obtained. These solutions are represented in Figure 9 by grey circles. Note that these dominated solutions are much closer to the efficiency frontier as compared with the published and peak timetables.

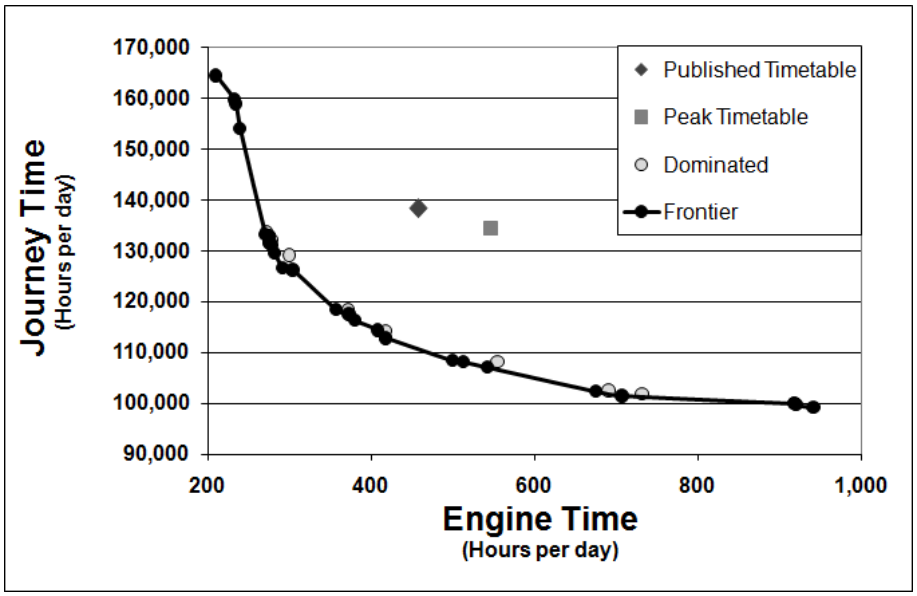


Figure 9: Approximate efficiency frontier

The Ayalon corridor is a 7 kilometer section connecting the Tel-Aviv stations; all segments in this section consist of three rails. This corridor is considered by the railway planners as the system's "bottleneck". For the $ratio = 0$ instance, the CE algorithm was able to schedule 32 trains per hour as compared with 20 trains in the peak hours of the current timetable.

Currently, Israel Railways is promoting a 2 billion NIS (approx. 500 million dollars) project for the development of a fourth rail in this corridor (http://www.haaretz.com/hasen/spages/685191.html). It appears that higher utilization of this corridor can be achieved without adding another rail. The algorithm can be used to test the effect of adding another rail in the corridor or, alternatively, the effect of increasing the capacity of some stations along the corridor.

5.5. **Further Validation**

In order to examine the behavior of the algorithm further, additional instances were created by modifying the routes input. The rest of the data for these instances are as in the original experiment. The additional instances are based on the two following route sets:

Route Set A: Uses existing infrastructure that is currently used only for freight transportation. In particular, several routes that bypass the Ayalon corridor were added.

Route Set B: Uses the original infrastructure but with different routes. All routes that pass through the Ayalon corridor were split into routes that begin\end at the corridor entrances. In addition, a high-frequency shuttle running along the corridor was added. Both route sets are potentially beneficial alternatives to the current routes because they allow easing the capacity constraint on the Ayalon corridor bottleneck.

A convergence test was conducted as in subsection 5.2. For each of the three tested weight ratios we used only the best algorithm settings as concluded from the experiment in subsection 5.1. Each instance was solved four times by the CE algorithm with a time limit of six hours per run. The average convergence results are reported in Table 7. The first two columns of the table describe the instance in terms of route set and cost ratio. Next, we present the difference between the average solution obtained by the CE algorithm and a solution obtained by a naïve (brute force) random algorithm after six hours as described in subsection 5.3. In the next two columns we present the average time in which the best solution was encountered by the algorithm and the time in which a solution within 1% of it was first obtained. In the rest of the columns we present the quality of the solution and the time when the no-improvement stopping criterion was first triggered for $\zeta = 15,30,50$.

Table 7: Convergence test for the additional instances

Run Setting		Brute force result	Initially Encountered (seconds)		No-improvement Stopping Criterion					
					15 Generations		30 Generations		50 Generations	
Set	**Ratio**	**% from best**	**Best solution**	**1% from best**	**% from best**	**Time (seconds)**	**% from best**	**Time (seconds)**	**% from best**	**Time (seconds)**
A	0	21.90%	21,632	16,837	6.4850%	7,374	-	-	-	-
	100	23.43%	15,409	7,719	4.7212%	8,937	0.0390%	13,459	0.0390%	14,309
	200	32.47%	12,329	5,767	0.1827%	6,938	0.1635%	7,684	0.1571%	9,144
B	0	32.72%	20,294	7,962	0.0340%	13,553	-	-	-	-
	100	23.79%	19,401	8,739	8.3003%	7,334	0.0590%	13,592	0.0199%	16,421
	200	30.26%	9,975	6,702	0.0699%	8,387	0.0061%	10,364	0.0000%	11,838

It is apparent from Table 7 that the algorithm delivers solutions that are significantly better than solutions that could be obtained by a brute force procedure. For the $ratio = 100,200$ instances, the algorithm converged within the six-hour time limit. This convergence pattern is similar to the one observed in the original experiment. The effectiveness of the no-improvement stopping criterion with $\zeta = 30$ or $\zeta = 50$ is supported by the results of this additional experiment.

6. Extensions

In this section, we describe possible extensions of the problem and discuss how to adapt the CE algorithm for them. In reality, the line planning and timetabling problem is typically solved incrementally. Several trains are added or modified while the rest of the system is kept unchanged. The unchanged trains constitute constraints on the occupancy of the infrastructure. It is straightforward to implement such constraints in the proposed framework: the insertion algorithm (see 4.2.1) is initiated with all the blocks and stations seized by the unchanged trains. The rest of the procedure remains the same.

Generally, it is desirable to set the frequencies of each inbound - outbound pair of lines to be the same. It is possible to implement this requirement by defining all routes as round-trip ones. This assures that whenever a train is scheduled, another one with an opposite route is also scheduled.

The demand for journeys is typically inhomogeneous throughout the planning horizon and is usually characterized by rush hours and slack hours. Thus, it may be possible to save on operational cost by reducing the frequency of some routes in slack hours without a significant increase in the total journey time. The same encoding method introduced in subsection 4.1 can be used to select some trains to be removed in these hours. This can be accomplished by adding another Boolean variable to each gene, representing whether the train is used in slack hours or not.

Throughout the paper, it is assumed that the capacity of trains is nonbinding and that all the trains are identical in terms of operational costs. It is possible to somewhat relax these assumptions by using the following model: the capacity of each train is controlled by the planner, and it determines the train's operational cost. To adapt the algorithm to this model, it is necessary to find the maximum occupancy of each train. The number of passengers on board a train at each block is calculated by assigning each passenger to his optimal itinerary, as in 4.2.3. The operational cost of each train is then determined by the maximum occupancy.

7. Conclusion

The calculation of the total time spent by passengers in the system is computationally involved. Hence, optimizing a line plan and timetable according to this measure is

challenging. The cross-entropy algorithm presented in this paper is shown to be effective in solving the problem. In addition, the ability of CE to tackle problems with complicated structures allows easy extension of the problem and incorporation of additional components of the rail planning process.

Acknowledgment: The authors are grateful to Mr. Joseph Navon, Mr. Yuri Morozov and Mr. Michael Shachar from Israel Railways for their assistance in defining the complicate real-life planning problem studied in this paper and for the data that they provided. The study was partly supported by the Government Companies Authority of Israel. The authors would like to thank the anonymous referees for their valuable comments and suggestions.

References

Abadi, I.N.K., N.G. Hall and C. Sriskandarajah. 2000. "Minimizing cycle time in a blocking flowshop". *Operations Research*, 48(1), 177-180.

Bendavid, I. and B. Golany. 2009. "Setting gates for activities in the stochastic project scheduling problem through the cross entropy methodology". *Annals of Operations Research*, 172(1), 259-276.

Borndörfer, R., M. Grötschel, and M.E. Pfetsch. 2007. "A column generation approach to line planning in public transport". *Transportation Science*, 41(1), 123–132.

Bouma, A. and C. Oltrogge. 1994. "Linienplanung und simulation für öffentliche verkehswege in praxis und theorie". *Eisenbahntechnische Rundschau* 43(6):369–378 (in German)

Bussieck, M.R., P. Kreuzer and U.T. Zimmermann. 1996. "Optimal Lines for Railway Systems", *European Journal of Operational Research*, 96, 54–63.

Bussieck, M.R., T. Winter and U.T. Zimmermann. 1997. "Discrete optimization in public rail transport", *Math. Program.,* vol 79, pp. 415–444.

Cacchiani, V. and P. Toth. 2011. "Nominal and Robust Train Timetabling Problems", *European Journal of Operational Research*, doi:10.1016/j.ejor.2011.11.003.

Caprara, A., L. Kroon, M. Monaci, M. Peeters, P. Toth. 2006. "Passenger railway optimization." In: Barnhart C, Laporte G (eds) Handbooks in operations research and management science, vol 14. Elsevier, Amsterdam, 129–187.

Ceder, A. and Y. Israeli. 1992. "Scheduling considerations in designing transit routes at the network level". M. Desrochers, J.-M. Rousseau, eds. *Proc. 5th Internat. Workshop Comput.-Aided Scheduling Public Transport* (CASPT), *Montréal, Canada, 1990*, Vol. 386. *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag, Berlin, Heidelberg, Germany, 113-136.

Chepuri, K., and T. Homem-de Mello. 2005. “Solving the vehicle routing problem with stochastic demands using the cross-entropy method”. *Annals of Operations Research* 134 (1): 153–181.

Claessens, M.T., N.M. van Dijk and P.J. Zwaneveld. 1998. “Cost optimal allocation of rail passenger lines”, *European Journal of Operational Research*, 110 (3), 474-489.

Cordeau, J.F., P. Toth and D. Vigo. 1998. “A survey of optimization models for train routing and scheduling”. *Transportation Science*, 32 (4), 380–404.

Cormen, T.H., C.E. Leiserson, R.L. Rivest, and C. Stein. 2001. “Introduction to Algorithms”, 2nd ed. (MIT Press, Cambridge, MA).

Goossens, J.H.M., C.P.M. van Hoesel and L.G. Kroon. 2004. “A branch-and-cut approach for solving railway line-planning problems”, *Transportation Science* 38, 379–393.

Gorman, M.F. 1998. “An application of Genetic and Tabu Searches to the Freight Railroad Operating Plan Problem.” *Annals of Operations Research, 78*, 51–69.

Kinder, M. 2008. “Models for periodic timetabling”. Master’s thesis, Technische Universität Berlin.

Liebchen, C. and R.H. Möhring. 2002. “A case study in periodic timetabling”. In *Electronic Notes in Theoretical Computer Science* Vol. 66 (6), pp. 1–14.

Liebchen, C. and R.H. Möhring. 2007. “The modeling power of the periodic event scheduling problem: railway timetables - and beyond”. In *Algorithmic Methods for Railway Optimization*, number 4359 in Lecture Notes on Computer Science. Springer, pages 3-40.

Lindner, T. 2000. “Train Schedule Optimization in Public Rail Transport”. PhD thesis, Technische Universität Braunschweig.

Michaelis, M and A. Schöbel. 2009. “Integrating line planning, timetabling, and vehicle scheduling: a customer oriented approach”. *Public Transp* 1(3): 211-232.

Müller-Hannemann, M., F. Schulz, D. Wagner, and C. Zaroliagis. 2007. “Timetable information: Models and algorithms”. In *Algorithmic Methods for Railway Optimization*, volume 4395 of LNCS, Springer, Heidelberg, pages 67-89.

Odijk, M.A. 1996. “A constraint generation algorithm for the construction of periodic railway timetables”, *Transportation Research.* Part B 30 (6) 455-464.

Rubinstein, R.Y. 1999. “The Cross-Entropy Method for Combinatorial and Continuous Optimization.” *Methodology and Computing in Applied Probability* 2, 127–190.

Rubinstein, R.Y. and D.P. Kroese. 2004. The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning, Springer-Verlag, New York.

Schmidt, M and A. Schöbel. 2010. "The Complexity of Integrating Routing Decisions in Public Transportation Models". *In Proceedings of 10th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*, ATMOS'10.

Schöbel, A. and S. Scholl. 2006. "Line planning with minimal travel time". In *5th Workshop on Algorithmic Methods and Models for Optimization of Railways*, number 06901 in Dagstuhl Seminar Proceedings, 2006.

Schöbel, A. 2011. "Line planning in public transportation: models and methods". to appear in OR Spectrum (available online).

Serafini, P. and W. Ukovich. 1989. "A mathematical model for periodic event scheduling problems", *SIAM Journal on Discrete Mathematics* 2, 550–581.

Törnquist, J. 2006. "Computer-based decision support for railway traffic scheduling and dispatching: a review of models and algorithms". In *Proc. 5th Workshop on Algorithmic Methods and Models for Optimization of Railways*, ATMOS 2005.

Wong R.C.W, T.W.Y. Yuen, K.W. Fung, and J.M.Y. Leung. 2008. "Optimizing Timetable Synchronization for Rail Mass Transit", *Transportation Science*, 42(1), 57-69.

Appendix A – Infrastructure representation of the benchmark problem