Service Oriented Train Timetabling

A thesis submitted toward the degree of Master of Science in Industrial Engineering

By

Mor Kaspi

May 2010
Service Oriented Train Timetabling

A thesis submitted toward the degree of Master of Science in Industrial Engineering

By

Mor Kaspi

This research was carried out in the Department of Industrial Engineering under the supervision of Dr. Tal Raviv

May 2010
**Acknowledgements**

To my advisor Dr. Tal Raviv, I wish to express my deepest gratitude for so many long guidance hours, for his endless patience and for his support during the hard moments of this work. Tal, along the road, you have taught me so much about the essence of academic research and the meaning of being a researcher. After quite a few mutual coffee brakes and piles of cookies, the work has come to an end. Thanks for everything!

I thank Michael Shachar, responsible for public applications in Israel Railways, for his willingness to help and his assistance in crossing bureaucratic obstacles. Additional thanks to Yosef Navon and Yuri Morozov of the Operational Planning Department in Israel Railways for the data and knowledge they were willing to share.

Thanks to Bette Lewis for proofreading this document and thanks to Iair Arcavi for the photographs of the Israeli Railways.

To my friends, thank you for helping me clear my mind at times that it appeared as if this thesis was taking control over me, for listening and for the various comments and suggestions.

The courses and research would have looked different without the presence of my academic partner and friend, Zohar Postelnick. Thanks for always taking care of administrative issues, the mutual lunches and for so many funny moments.

Finally, thanks to my entire family for supporting me throughout this study. Especially to my parents for allowing me to conveniently focus on my studies and for knowing when and how to push me ahead, and to my twin brother Ran who spent many hours listening to me talk about this subject.

*This research was awarded financial support by the scholarship program of the Government Companies Authority.*
"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

The train planning problem can be divided into several sub-problems, mainly Line Planning Timetabling, Platforming, Rolling Stock Circulation and Crew Planning. In this study, we focus on quality of service aspects derived by decisions made during the line planning and timetabling phases. The quality of service is measured in terms of total time spent by the passengers in the railway system, including waiting time at the stations of origin, connections and travel time. We formulate an integrated Line planning and Train Timetabling model, devise methods to encode a feasible solution and to quickly evaluate it. We then apply the Cross Entropy meta-heuristic in order to solve the problem. We use the Israeli railway system as a test bed. The timetable created by our algorithm saves about 20% of the total time spent by the passengers in the system, compared to the one currently in use. Similar methods are used to solve a bi-objective problem that considers both operational costs and service level. Pareto dominating solutions are derived.
# List of Contents

1. Introduction ................................................................................. 1

2. Definitions .................................................................................. 2
   2.1. Infrastructure Related Definitions ........................................ 2
   2.2. Railway Planning Definitions ............................................... 4

3. Literature Review ...................................................................... 6
   3.1. Line Planning Problem .......................................................... 6
   3.2. Train Timetabling Problem ................................................... 8
      3.2.1. Non-Cyclic Timetabling ................................................ 8
      3.2.2. Cyclic Timetabling ....................................................... 12
   3.3. Other Sub-Problems .............................................................. 14
      3.3.1. Platforming ................................................................. 14
      3.3.2. Rolling Stock Circulation .......................................... 14
      3.3.3. Crew Planning .......................................................... 14
   3.4. Integrating Line Planning and Train Timetabling .................. 15
   3.5. Literature Gap ..................................................................... 16
   3.6. Cross Entropy ...................................................................... 17
      3.6.1. Rare-Event Simulation ................................................ 17
      3.6.2. Solving Combinatorial Problems with the CE Method .... 18

4. Problem Definition .................................................................... 20
   4.1. Sample problem .................................................................. 23
      4.1.1. Input ........................................................................... 23
      4.1.2. Solution ....................................................................... 25

5. Methodology ............................................................................. 27
   5.1. Solution Encoding ............................................................... 28
5.2. Decoding One Cycle. ................................................. 29
  5.2.1. Insertion Process - Block Seizing and Station Seizing .......... 30
  5.2.2. Sample Problem .................................................. 32
  5.2.3. Sequence of Blocks. ............................................. 35
  5.2.4. Traversing Blocks in Minimal time .............................. 36
5.3. Constructing a Timetable ........................................... 39
5.4. Timetable Evaluation ................................................ 42
  5.4.1. Reachability Algorithm. .......................................... 43
  5.4.2. Calculation of the Objective Function Value ..................... 45
5.5. Cross Entropy Application .......................................... 46
6. Numerical Experiments .................................................. 50
  6.1. Benchmark Problem. ............................................... 51
  6.2. Experiment Parameters ............................................ 54
6.3. Statistical Analysis of Parameters Tuning Experiment ............ 54
  6.3.1. First Experiment ................................................ 54
  6.3.2. Second Experiment. ............................................. 57
  6.3.3. Convergence of the Algorithm .................................. 60
  6.3.4. Comparing CE with Brute Force Approach ...................... 62
6.4. Comparison to Current Timetable. ................................ 63
7. Conclusions. .................................................................. 66
  7.1. Main Contribution of Study ........................................ 66
  7.2. Recommendations for Future Research .............................. 68
References ....................................................................... 70
List of Figures

Figure 1: Illustration of a small network .................................................. 3
Figure 2: Notation for railway infrastructure. ............................................ 3
Figure 3: A train traveling through a station .............................................. 4
Figure 4: Time-Distance diagram .............................................................. 5
Figure 5: A graph representation of the network displayed in Figure 1 ........ 20
Figure 6: Minimal headway ................................................................. 21
Figure 7: Sample problem - A graph representation of the network, including
minimal traveling times on the blocks and stations capacities. ................. 23
Figure 8: A solution for sample problem – Time-Distance diagram. .......... 25
Figure 9: A solution for sample problem – blocks display ......................... 25
Figure 10: A gene. ........................................................................... 28
Figure 11: A part of a chromosome ....................................................... 29
Figure 12: A solution for sample problem – insertion of train 10 .......... 33
Figure 13: A solution for sample problem – insertion of train 9 .......... 33
Figure 14: A solution for sample problem – insertion of train 3 .......... 34
Figure 15: A solution for sample problem – insertion of train 4 .......... 34
Figure 16: A solution for sample problem – insertion of train 12 .......... 34
Figure 17: A solution for sample problem - insertion of train 11 .......... 35
Figure 18: Infrastructure parts that satisfy and do not satisfy Assumption 3 .... 36
Figure 19: A solution for sample problem – a part of the events graph .... 42
Figure 20: A passenger planning her journey ......................................... 45
Figure 21: Infrastructure representation of the benchmark problem .......... 52
Figure 22: Convergence of the best solution in the second experiment .... 60
Figure 23: Efficiency frontiers .................................................................. 65
Figure 24: A solution for sample problem – scheduling of one cycle .... 68
List of Tables

Table 1: Estimation of passengers demands on a weekday, between 7am-8am (2008) ................................................................. 4
Table 2: Sample problem – origin-destination matrix ................................................. 24
Table 3: Sample problem – additional problem parameters ........................................ 24
Table 4: A Solution for sample problem – the timetable .............................................. 26
Table 5: A solution for sample problem – chromosome representation ..................... 32
Table 6: A solution for sample problem – sorted list of trains to be inserted ............... 32
Table 7: Spreading the events of train 11 ................................................................. 40
Table 8: Duplicating the events of train 11 (trains 111 and 211) ................................. 41
Table 9: Parameters of the benchmark problem ....................................................... 53
Table 10: First experiment results .......................................................... 55
Table 11: First experiment – Effects estimation ......................................................... 56
Table 12: First experiment – ANOVA ................................................................. 56
Table 13: Second experiment results .................................................................. 58
Table 14: Second experiment – Effects estimation .................................................... 59
Table 15: Second experiment – ANOVA ............................................................... 59
Table 16: Solutions progress ................................................................................. 61
Table 17: Results of brute force runs .................................................................. 62
Table 18: Comparison to current timetable – timetables values .............................. 63
Table 19: Comparison to current timetable – trains per hour on Ayalon corridor 63
1 Introduction

Railway operations planning has been practiced since the first trains started working early in the nineteenth century. The Planning problem consists of several interrelated sub-problems, namely

- Line Planning – deciding which set of lines should be served by the system and in what frequencies, subject to total demand for journeys and capacity constraints of trains and of the infrastructure
- Train Timetabling – deciding upon the schedule of each train in each line subject to track availability and headway constraints.
- Platforming - assigning a platform to each train that is routed through a station, assuring safe entrance and departure from the station.
- Rolling Stock Circulation – assigning cars and locomotives to each scheduled train.
- Crew Planning (rostering) - assigning crews to all scheduled trains.

Each of these sub-problems is computationally difficult.

Although in practice, some of the planning work is still being done manually, in the last 40 years planners have been using Decision Support Systems and optimization methods in order to improve the quality of their plans and to save labor. Typically, the planning problem is solved hierarchically in a strategic order, such that the solution of each sub-problem is used as input for the following problems. This solution strategy enables tackling real-world problems; the disadvantage is that the global optimal solution is lost "on the way."

In this study we will integrate the Line Planning Problem with the Train Timetabling Problem in order to create Line Plans and Timetables with better quality of service. We view quality of service as the total time spent by passengers in the railway system. More explicitly, the total travel time of all passengers includes waiting times at origin station, time on board of the trains and transfer times.

Methods to encode the railway infrastructure and a feasible solution for the integrated problem will be presented and the Cross-entropy meta-heuristic will be applied in search for good solutions.

The thesis is organized as follow: In Chapter 2 we define some basic concepts of railway planning. In Chapter 3 we review the literature on railway operations’ planning optimization models. A gap in the state of the art to be addressed by this study is identified. In addition, we give a review on the Cross-Entropy method. In Chapter 4 the Line Planning and Train Timetabling Problem is defined. In Chapter 5 we lay out the algorithm used to solve the problem. In Chapter 6 we report results of a numerical study based on actual data from the Israeli railway system. Chapter 7 contains conclusions and recommendations for future research.
2 Definitions

In this chapter we introduce the terminology used in railway operations planning. We note that different studies in this field may use different terminology. Therefore, we first define some standard concepts in railway planning. The literature review in the subsequent chapter makes use of our standardized terminology rather than the original one used by the various authors.

2.1 Infrastructure Related Definitions

Block (Track Section) - a part of the track. The track is usually divided into blocks where each block can normally hold one train at a time in order to maintain the required safety level.
Segment - composed of one or several parallel blocks.
Passenger Station - A special segment where trains load and unload passengers.
Siding - A special block that can be used for the crossing and passing of trains. Sidings are typically found in stations but can be found in any segment.
Signal - a light or a semaphore on a railway, giving indications to train drivers of whether or not to proceed. Signals are used to break long track sections into several blocks in order to allow better utilization of the infrastructure.

Route - A sequence of blocks, stations and sidings between two stations that may be regularly traversed by a train.
Line - A sequence of segments, stations and sidings between two stations. A published timetable refers to lines (the specific routes used are transparent to the passengers). More specifically, a line is characterized by its origin and destination station, the sequence of segments and the intermediate stops at passing stations.
Slot - A tuple consisting of a block and a time interval in which the block is traversed by a train.

Network - A set of lines, possibly with common segments.

(Scheduled) Train - a sequence of slots embedded in a route and consistent with the travel times of the rolling equipment on the corresponding blocks.

Traffic direction – the traffic on each block can be restricted to one direction (unidirectional) or can be permitted on both directions (bidirectional). Typically, in segments that consist of two parallel blocks, each block is unidirectional and the direction of the two blocks is opposite.
Figure 1: Illustration of a small network.

Figure 1 illustrates a small network. The network contains 7 segments, 5 stations and one siding. The segment between station A and the siding contains one block on which the traffic is allowed on both directions (bidirectional). The segment between stations B and C contains two parallel blocks, on each block the traffic is allowed in only one direction (unidirectional).

In her review, Törmquist (2006), introduced a notation for railway infrastructure that refers to three important aspects of it, namely, network vs. single line model, traffic direction and number of parallel tracks both in regular segments and in stations. A template of this notation is presented in Figure 2 below. A railway system is said to be bidirectional if at least one of its segments is bidirectional. The infrastructure studied in this thesis is the most general one, namely (N)(B:BN)(S:BN).

Figure 2: Notation for railway infrastructure
2.2 Railway Planning Definitions

**Train conflict** – A situation in which more than one train occupies the same block. Conflicts may occur when two trains approach each other and when a train catches up with another train traveling in the same direction.

**Minimum Headway** – The minimal time required between two trains. This rule is used as a safety measure to deter train conflicts. For trains traveling in the same direction: since only one train is allowed to seize a block at a time, the traveling time on the block serves as this minimal time. For trains traveling in opposite directions, an extra time interval is required.

**Transfer** – Disembarking one train and later embarking on another in order to complete a journey. This action may require changing platforms in the transfer station.

![Figure 3: A train traveling through a station](image)

**Origin-Destination (O-D) matrix** – represents the passengers' demands for journeys between each origin-destination pair starting in a certain time period. For example, according to Table 1, the number of passengers that arrive at Haifa station between 7am-8am and wish to travel to Be’er Sheva is 46.

<table>
<thead>
<tr>
<th></th>
<th>Haifa</th>
<th>Tel – Aviv</th>
<th>Be'er-Sheva</th>
</tr>
</thead>
<tbody>
<tr>
<td>Haifa</td>
<td></td>
<td>887</td>
<td>46</td>
</tr>
<tr>
<td>Tel – Aviv</td>
<td>475</td>
<td></td>
<td>216</td>
</tr>
<tr>
<td>Be'er-Sheva</td>
<td>43</td>
<td>349</td>
<td></td>
</tr>
</tbody>
</table>

**Table 1**: Estimation of passengers demands on a weekday, between 7am-8am (2008).
**Cyclic Planning** - Planning approach in which every planned task or event is repeated exactly for every fixed time period, which is called cycle time. In this approach, only one time period has to be planned and validated. This planned time period is then repeated over the planning horizon. A typical cycle time in a passenger railway system is one hour. Cyclic schedules are very common in railway systems because it simplifies the design process and can be conveniently communicated to the passengers.

**Line Frequency** – Indicates the number of trains that are scheduled to use the line in a certain time period, usually one hour.

**Time distance diagram** – A diagram that is used by planners to validate a given schedule, i.e. to make sure that there are no planned train conflicts. The distance along a specific route is shown on the vertical axis, and time on the horizontal. The path of a train is represented by a line whose angle of inclination to the horizontal increases with speed, and in which an intermediate stop is represented by a short horizontal line.

![Time-Distance diagram](image)

**Figure 4:** Time-Distance diagram

In Figure 4, trains 1 and 3 are planned to travel from station A to station D with an intermediate stop at station B. Trains 2 and 4 are planned to travel from station D to station A with intermediate stops at stations C and B. Trains are only planned to meet at stations, there they can safely pass each other, using different platforms. To verify that there are no conflicts in this plan, check that at any moment one train at most uses each block and that a certain time interval separates trains traveling in opposite directions on the same block.
3 Literature Review

In this chapter we present a review on the state of the art of the literature on railway operations planning and identify the gap to be addressed by this study. For survey papers on various optimization problems encountered by railway planners, see Bussieck et al. (1997), Cordeau et al. (1998), Caprara et al. (2006) and Törnquist (2006).

In addition, we present the Cross Entropy (CE) meta-heuristic which was proposed by Rubinstein (1997) and is applied in this study.

3.1 Line Planning Problem

The line planning problem is defined as follows: for a given railway infrastructure (given as a set of interconnected blocks and stations), traveling times on each block and demand represented by O-D matrices, determine the set of lines and frequencies so as to satisfy some operational constraints, such as demand satisfaction, and optimize some operational goal, such as costs and service levels. The literature on this problem is relatively limited. One reason for that may be the fact that planners seldom face a new line planning problem. Significant changes in the infrastructure may take years or even decades, the same is true for significant changes in the passengers' demands. Changes in the line plan tend to be incremental and may be heavily influenced by stakeholders, such as politicians and others.

Many large scale railway systems can be reasonably split into several subsystems. Namely, national systems consist of high speed and high capacity trains that typically stop only at major stations, and several regional and metropolitan subsystems that make use of slower rolling equipment and stop more frequently. This phenomenon is referred to as system split. Under the system split assumption, a traveler will transfer from a slower train to a faster train at the earliest possible point of his journey and, vice versa, will leave the faster train as late as possible, if it is necessary to travel in a slower train at the end of the journey. This assumption determines the distribution of passenger flows (traffic assignment) to the different segments. Unless explicitly stated, all the studies reviewed below made the system split assumption.

Bussieck et al. (1996) present a MIP model for choosing a set of lines and their frequencies so as to maximize the number of direct travelers. The authors assume that passengers between any two stations use a shortest path with respect to travel time or travel distance. The lines are chosen so that on each segment a predetermined frequency of trains is ensured. The model does not take into account the operational costs of the lines. These costs are somewhat bounded by the frequency constraints, namely, the total number of trains that can be run is bounded. The direct-travelers variables represent the number of direct travelers between each pair of stations on each line, this leads to a huge model which is unsolvable for real-world instances. The model size is reduced significantly by changing these variables to represent only the number of direct travelers between each pair of stations, without taking into account what line is used (assuming that the distribution of travelers to lines is carried out later). The reduced model is solvable in a reasonable time after introducing some cutting planes. The solution of the reduced model is a lower bound on the optimal solution of the original problem. In addition, a feasible solution for the original
problem can be derived from this optimal solution. For all examined instances, the optimality gap was less than 3.2%.

Claessens et al. (1998) solve a line planning problem integrated with decisions on trains’ capacities with the objective of minimizing the operating costs. First, they prove that the problem they face is NP-hard by a reduction from the vertex cover problem. Later, a non-linear mixed integer model is formulated. The model's constraints ensure that all passengers are served, namely that the frequency of the lines is in between certain bounds and the number of cars used in each train is also bounded. This initial model is changed into a tight Binary LP model so that it can be solved with a commercial solver.

Gossens et al. (2004) use a minimum cost model similar to the model formulated by Claessens et al. (1998). They introduce several new cuts and apply a branch and cut algorithm to solve the problem. In addition, they introduce preprocessing techniques for the elimination of coefficients, variables and constraints.

Schöbel and Scholl (2005) present a service oriented approach for the Line Planning with the objective of minimizing total travel time (net of waiting time at the stations). A directed graph with nodes representing (station, line) tuples and arcs representing lines and possible transfers between lines is introduced. The length of the arcs represents the travel times and predefined transfer times. This graph is called a change-and-go network. Given an O-D matrix, passengers’ traffic is assigned to lines, assuming each passenger chooses a journey that minimizes her total net travel time. An Integer LP is formulated to decide upon line frequencies so as to minimize total net travel time of all passengers subject to a budget and passenger capacity constraint. The LP-relaxation of the model is solved using a Dantzig-Wolfe decomposition.

Borndörfer et al. (2007) propose a multicommodity flow model that has two competing objectives: minimizing the operating costs and minimizing the net passenger traveling time (on-board time). The underlying assumption is that vehicle travel time and passenger delay have known costs so that the objectives compete in the same terms. The model does not assume system split, so that the passengers' paths can be freely determined after the lines are computed. The model is formulated as an integer program with numerous variables and a column-generation approach is taken in order to solve the LP relaxation. The authors prove that if all computed lines are restricted to logarithmic length, with respect to the number of stations in the network, then the column generation problem can be solved in polynomial time.
3.2 Train Timetabling Problem

The train timetabling problem is defined as follows: for a given set of lines, frequencies and possibly departure times of each train, determine the arrival and departure time at each block and station such that safety constraints are satisfied and possibly other goals are achieved. The literature on train timetabling is abundant compared to other aspects of the railway planning process. Some of the literature focuses on cyclic timetables while other studies focus on non cyclic ones. Non cyclic planning is mainly used in freight trains scheduling where the demand for trains is irregular and may be unknown in advance and the travel time of each train may vary from a few hours to a few days. The planner receives requests for trains for the entire planning horizon and has to try to schedule as many trains as possible or maximize the profit for the railway company (in cases where each train represents a different potential profit).

Cyclic Timetabling is mainly used in passengers' trains scheduling where passengers' demands can be estimated far in advance, journey times vary between a few minutes to a few hours and the frequencies of lines are relatively high. In a cyclic timetable, each trip is operated repeatedly once or a few times during each cycle. From the passengers' point of view, such timetables are more convenient because all they need to remember are the times in a cycle when trains arrive at their station. From the planners' point of view, since each event occurs over again in every cycle, it is enough to plan one cycle, thus reducing the search space significantly.

3.2.1 Non-Cyclic Timetabling

Szpiigel (1973) considers infrastructure of type \((N)(B:BS)(S:B\infty)\) and solves a basic scheduling problem on a single track railroad. The model's objective is to find the best crossing and overtaking locations of trains given their routes and departure times so that weighted average train travel time is minimized. Szpiigel shows that the problem is analogous to the general job-shop scheduling problem and uses a branch and bound algorithm to solve it. Branching is carried out by selecting two trains and fixing their meeting segment. Computational experiments are presented for a system with five segments and ten trains in eastern Brazil.

Jovanovic and Harker (1991) \((L)(B:BD)(S:BD)\) present a decision support system called SCAN. The purpose of SCAN is to help in the design of feasible timetables, rather than to provide an optimal solution for the Train Timetabling Problem. The model evaluates the feasibility of a given set of scheduled trains, if this set is found to be infeasible, i.e., overlapping slots are used, the system offers interactive or automatic procedures to modify the timetable until it becomes feasible. The system does not try to build a new timetable from scratch. This method was chosen due to the fact that most planners are reluctant to completely throw away their initial timetable. The authors define meetpoints as any points along the line, including the end stations where trains can wait in order to meet with each other, overtake or be overtaken. Meetpoints include stations, sidings and connection points of double track segments and single track segments. A meet-pass plan is defined in terms of the locations along the given line where each interacting pair meets or overtakes; a feasible meet-pass plan is one that ensures that all trains can be inserted. The problem of generating
feasible meet-pass plans can be viewed as a mixed-integer programming problem without an explicit objective function and the goal is to find all or a pre-specified number of feasible solutions. The decision variables for this problem are integers which represent the number of the meet-points used for meeting or passing of a particular train pair, and real-valued variables which represent the arrival and departure times of the trains at/from the various meet points. A variation of the meet-pass plan generation algorithm was developed and implemented in the SCAN model especially for the purpose of assisting the scheduler/analyst with determining sources of infeasibility and ways to derive a feasible schedule from an infeasible one.

Carey and Lockwood (1995) \((L)(B:US)(S:U\infty)\) set out a model and algorithms for train timetabling. The basic model considers a line with multiple stations and multiple trains of different speeds, with all trains on the line traveling in the same direction, as the case in the European railways system. Carey (1994) \((N)(B:BN)(S:B\infty)\) extends the model to a more general network, introducing explicit choices among N-block segments, choice of platforms to use at stations, shared platforms, various types of intersections and trains in one or both directions, etc.

A binary integer program is introduced. The binary variables represent the precedence order of each pair of trains on each block. The objective function is based on a cost function of the delays. To solve the problem the authors propose adopting strategies analogous to those adopted by "expert" train pathers using traditional manual graphical methods (most of them are later translated into cutting planes). The basic strategy is to insert the trains one at a time until all trains are scheduled, and if necessary, iteratively repeat trains until an acceptable solution is found. In this manner, a sub-problem is created, which is to insert a train given the slots of the already scheduled trains. A major advantage of this strategy is that although the numbers of binary variables increases if the problem size is increased by introducing more trains, the number of variables in the sub-problems does not increase.

Carey and Lockwood (1995) also introduced a number of other complementary strategies to further reduce model dimension and its computing time needed to solve the integer program. These strategies are based on analogies with how experienced or expert train planners’ schedule trains, using traditional manual graphical methods. Reducing the search space:

1. **Time windows.** Placing bounds on train time slots enables the elimination of 0-1 variables and constraints.
2. **Fixing train slots.** In the sub-problem the times of all trains vary within bounds. However, the timetable of any train whose time window has no overlap with that of the current train can be fixed.
3. **Dynamic recalculation of bounds.** Fixing the precedence order of a train (and those of other trains) is likely to restrict the range within its time slots can vary.
4. **Preprocessing bounds to eliminate 0-1 variables and constraints.** The 0-1 variable in the sub-program is zero and can be eliminated if there is no sufficient "space" (time) between two consecutive trains to insert the train. Even if one train departs as early as possible and the next train departs as late as possible, there may not be sufficient room between them to allow the headways needed to insert the new train.
5. **Preventing repassing-** a pair of trains can only meet once along the line.
6. **Flighting trains.** This refers to the common practice of dispatching trains in groups having similar speeds. High-speed trains will be scheduled first then slower ones.

Using smart branching rule

1. **Searching routes in block traversal order.** When branching in the branch and bound search, the 0-1 variables associated with choice of blocks are introduced in the same order as the blocks are traversed by the trains.

2. **Depth-first search.** Following initial experiments they found it best (much faster) to use a diving or depth first search strategy. It reflects the pather's concern with quickly finding "acceptable" feasible or viable solutions, rather than with global optimality.

The paper presents a Numerical example with a network of 5 stations and 28 single unidirectional segments.

Higgins et al. (1997) \((L)(B:BS)(S:UD)\) consider the problem of creating a timetable that minimizes the total train travel time. They apply enhanced local search heuristic, genetic algorithm, tabu search and two hybrid algorithms. A feasible solution is represented by the meeting points of all potentially interfering pairs of trains. When two trains are to interfere with each other (cross or overtake), the main decision variable for the train controller is to decide which train is to be delayed at a siding. A neighborhood is defined by the set of all solutions obtained by moving one such meeting point. The largest instance solved is of 50 trains and 113 conflicts.

Brännlund et al. (1998) \((N)(B:BN)(S:BN)\) present a mathematical programming model for timetabling on a single bidirectional track. The model's objective is to maximize profit, where each train has a profit function, depending on the time of departure from the terminal minus penalties for waiting along the track. A Lagrangian relaxation solution approach is used, in which the seizing constraints (that deter the presence of two trains on the same block at the same time) are relaxed and assigned prices, so that the problem separates into one dynamic program for each train. The number of dual variables is very large. However, only a small fraction of these are nonzero. The dual updating scheme takes advantage of this fact.

To begin with all trains to be scheduled are arranged in a priority list. The intercity trains have priority over regional passenger trains, which in turn have priority over freight trains. The motivation for this is that extra waiting time is usually not a big issue in the case of freight trains, whereas in the passenger case, it certainly is. In each class, trains with high values have priority over trains with lower values. Using the priority list, the trains are scheduled one after the other, using the current shadow prices for using the blocks. The highest priority train will get its most desired schedule. Thereafter, the space–time network for the next train in the list is modified by removing arcs representing already occupied blocks. The train is then scheduled by finding a profit maximizing path in the reduced network. The corresponding timetable is necessarily feasible with respect to the seizing constraints through the construction of the reduced network. The process is repeated until all trains have been scheduled. In this problem setting, it is not necessary to schedule all trains. Hence, the heuristic always generates a feasible solution. Two test instances based on a single line \([L](B:BS)(S:BS)\) of the Swedish railway system are presented, with up to 22 passenger trains and 8 freight trains to be scheduled on 16 single bidirectional segments connecting 17 stations.
Caprara et al. (2002) \((L)(B:US)(S: B\infty)\) model a single unidirectional line linking two major stations with a number of intermediate stations in between. The authors justify their interest in this particular case by the fact that in some railway networks one busy line (corridor) is the bottleneck of the whole system. Once the timetable for the trains on the corridors has been determined, it is relatively easy to find a convenient timetable for the trains on the other lines of the network. A graph theoretic representation of the problem is proposed using a directed multi-graph. Nodes correspond to departures/arrivals at a certain station at a given time instant and the connecting arcs represent optional slots. An integer linear programming model is created where binary variables are used for the representation of each arc in the graph. The objective is to maximize the sum of profits of scheduled trains. A Lagrangian relaxation approach is taken in order to solve the model. Because the number of arc constraints is exponentially large, the number of resulting dual variables makes the algorithm impractical. The authors then give a new formulation in which integer variables representing the graph nodes are added, with some relevant constraints. The advantage of this formulation is that the Lagrangian relaxation leads to a relaxed problem in which the profits of the arc variables are unchanged, whereas Lagrangian penalties are associated with the nodes which are much easier to handle. A dynamic constraint generation scheme (relax-and-cut) is used to handle the large number of relaxed constraints, explicitly considering each constraint only when it turns out to be violated by the relaxed solution at some iteration of the subgradient procedure. A computational study on seventeen instances with up to 73 stations and segments and up to 500 trains was performed. Approximated results with optimality gap of up to 20% were obtained. These instances were received from FS Rete Ferroviaria Italiana, the Italian railway infrastructure operator.

Zhou and Zhong (2005) \((N)(B:US)(S: B\infty)\) consider the train scheduling problem as a multi-mode flow-shop scheduling problem with multiple criteria. Specifically, a double-track rail corridor consists of a set of sections (machines) and a set of trains (jobs). Each train has a predefined route (processing order) through segments, and a train traveling along a route can be viewed as a procedure of completing a series of activities with respect to segments. To model acceleration and deceleration time losses, an activity is assumed to have multiple execution modes, corresponding to different running states (i.e. stop or no-stop) and durations. The first objective of this problem is to minimize the variation of inter-departure times for high-speed trains and the second objective is to minimize the total travel time of all trains. A branch-and-bound algorithm is developed providing an implicit enumeration method to identify complete non dominated solutions.

Recently, Cacchiani et al. (2008) \((L)(B:US)(S:U\infty)\) solved the same problem presented in Caprara et al. (2002). They formulated a different ILP model that represents each possible path for a train by a binary variable. A method to solve the LP relaxation is proposed, it combines column generation and separation techniques. Heuristic algorithms are then applied to create a solution for the ILP model from the optimal LP relaxation solution. The proposed method is compared to the previous Lagrangian method of Caprara et al. (2002) using the same problem instances. Better solutions could be obtained but at a significant computational cost. The lower bounds calculated by the heuristic algorithms are higher by up to 2% from those obtained by the Lagrangian heuristics, but with calculation time up to 80 times longer.
3.2.2 Cyclic Timetabling

Most of the literature concerning cyclic timetabling refers to mass-transit systems in which each line has its own rails and platforms and therefore some of the safety constraints introduced in Subsection 3.2.1 can be relaxed. As pointed out by Wong and Leung (2004), a common assumption in transfer-scheduling practice is that the capacity of the trains can receive all passengers who want to board the train. Ignoring the capacity constraints highly reduces the complexity of the problem. Indeed, in many situations capacity issues can be effectively addressed in line planning and rolling stock allocation phases.

Serafini and Ukovich (1989) propose a general mathematical model for scheduling activities of periodic type. The Periodic Event Scheduling Problem (PESP) is to set the times (in a period) of a given set of periodic events so as to satisfy a set of periodic constraints. Each periodic constraint represents the relationship between an ordered pair of periodic events. The events are scheduled for one period (cycle time) so that later this period can be repeated over the planning horizon. In addition to the general model, the authors present more specific cases on which the model can be applied such as: periodic job-shop, transportation scheduling and traffic light scheduling. The authors conclude with a suggestion to adapt the general model to specific problems in order to improve computational efficiency. Many of the studies on railway cyclic timetabling have adopted the PESP paradigm.

Nachtigall and Voget (1996) (N)(B:US)(S: B∞) create cyclic timetables for mass transit network. Each line in the network has a predefined sequence of stations which the trains traverse in a fixed time. Thus, scheduling a train is completely done by deciding on the departure time at the initial station. The given lines are scheduled one-by-one in order to form a timetable. The objective is to minimize the total waiting times of passengers at transfer stations. The writers present a greedy algorithm to choose the order of lines in the scheduling process. Later a genetic algorithm is used for decisions on the starting times of each train.

Goverde (1998) Considered a (N)(B:UD)(S:UD) underground system with no interring lines. The paper introduces the problem of determining how long connecting trains should wait for transferring passengers in case of unexpected delays of other trains. The objective is to minimize the total weighted waiting times of passengers on the delayed trains and of the transferring passengers. The author models the passenger waiting times by recursive equations in max-plus algebra.

Liebchen and Möhring (2002) (N)(B:US)(S:US) use the Berlin underground (BVG) as a case study. They use PESP to create feasible timetables with average passenger changing time as an objective. They introduce some criteria such as: the amount of rolling stock required, the average speed of trains and the number of cross-wise correspondences (ensuring good correspondences in every station where different lines share a platform). They iteratively add "political" constraints presented by the BVG planners and show that even with these constraints the performance of the optimized timetables is better than the performance of the manually constructed timetables.
Wong and Leung (2004) considered a \((N)(B:UD)(S:UD)\) underground system with no interring lines. They solve the problem of minimizing transfer waiting-times of all passengers. In their model the path of each passengers flow is predetermined, namely, it is assumed that scheduling decisions do not affect the passengers' paths. Coordination and synchronization of the schedule is achieved by adjusting the running times, dwelling times and dispatching times of each train in a cycle. The authors demonstrate that flexible dwelling times allow improving the average waiting times of passengers during transfers as compared to fixed schedules.

Kroon et al. (2005) \((L)(B:UN)(S:UN)\) formulate a stochastic PESP model in which random disturbances may occur. The goal is to supplement the running time of the trains along some blocks so as to minimize the average weighted delay of the trains. A timetable is evaluated by generating numerous realizations of the trains according to this timetable but under stochastic disturbances and by measuring the average delay in some stations. Next, the stochastic PESP instance is built by unfixing most of the event times and by taking into account the same predetermined disturbances. The model modifies the timetable by re-allocating the time supplements in such way that the average delay is minimized.
3.3 **Other Sub-Problems**

In this following section we briefly review literature related to other railway planning sub-problems

3.3.1 Platforming

The constructed timetables that result from the abovementioned problems ensure that the capacity constraints of the stations are not violated. However, the detailed structure of each station with its safety rules is usually not taken into account. The Train Platforming Problem objective is to find a suitable path in a station for each train that is scheduled to arrive and depart from the station. The infrastructure of a station is given as an input, including the internal blocks (rails and platforms) and the traveling direction allowed on each block. The main constraints ensure that at least a minimum time gap is kept between two consecutive trains that reach the same platform and that overlapping routes cannot be used in the same time.

Odijk (1996) applies a cutting plane algorithm based on PESP in order to construct a cyclic timetable in a major railway station in the Netherlands. The cycle time is 30 minutes; the station has six platforms and three incoming and outgoing directions. In each cycle 12 different trains visit the station. The created timetable satisfies 54 given constraints. Although the studied example is a Platforming problem, the presented algorithm can be used to solve a cyclic timetabling problem (Section 2.2.2). For other studies on the Train Platforming Problem, see Zwaneveld et al. (1996), Kroon et al. (1997) and Carey and Carville (2003).

3.3.2 Rolling Stock Circulation

For a given timetable we need to assign rolling equipment (train units or carriages) to all scheduled trains. The Rolling Stock Circulation Problem is to determine the optimal stock of rolling equipment needed to allow the operation of the whole timetable. The objective is to minimize the circulation costs, generally under constraints that assure a certain amount of rolling equipment for each scheduled train. See relevant chapters in surveys by Cordeau et al. (1998) and Caprara et al. (2006).

3.3.3 Crew Planning

Crew planning is the final stage of the train planning process. At this stage the planners hold a full operational timetable (including empty train journeys). The problem is to assign crews to all scheduled trains with minimum cost under some operational constraints. This problem is usually decomposed to two problems: Crew Scheduling and Crew Rostering. The scheduled trains are split into a sequence of trips defined as sections of train journeys which must be serviced by the same crew without rest. In the Crew Scheduling part, a set of duties covering all the trips is constructed. Each duty represents a sequence of trips to be covered by a single crew within a given time period. In the Crew Rostering part, the duties selected in the previous part are sequenced to obtain the final roster. See Caprara et al. (1997) for a survey of algorithms for railway crew management.
3.4 Integrating Line Planning and Train Timetabling

As mentioned earlier, there is a substantial potential for improvement by integrating some of the train planning sub-problems and solving them simultaneously. The following studies integrate the Line Planning Problem and the Train Timetabling Problem.

Gorman (1998) applies Genetic and Tabu searches to the freight railroad operating plan problem. The Operating Plan Problem consists of aspects of both the Line Planning Problem and The Train Timetabling Problem. More specifically, the problem is to decide upon the weekly frequencies of the trains and their stopping stations. The objective is to minimize operating costs subject to customers' expectations and congestion restrictions.

The study focuses on a major US freight railroad. A “menu” of the possible routes is specified in order to reduce search space. The time horizon, a week, is discretized into hours. Binary variables are used to encode a solution, each variable represents a certain route starting at a certain time of the week (1 = run train starting at this time, 0 = no train). In this setting, the issue of conflicting trains is approximated by a flow rate constraint per segment. A Genetic algorithm is used to search for good schedules. Every time a solution is generated, its cost is evaluated by solving a traffic-assignment problem. A generated schedule must service all given demands, infeasible schedules are penalized. A randomly chosen crossover site is chosen for the crossover point of the two parents. Mutations are obtained by either adding or deleting a train, or by shifting a train to an earlier or a later time in the schedule. To improve the performance of the genetic algorithm, each solution is cloned and modified with a Tabu search algorithm.

The tabu-enhanced genetic algorithm generated solutions for a full-scale operating plan problem (with 90 different trains) that reduce both operating costs and penalty costs compare to the schedule actually run by the railroad company.

Lindner (2000) (N)(B:BN)(S: B∞) formulates an integrated Line Planning and Train Timetabling mixed integer linear program that minimizes the scheduling cost. A branch and bound method for the Line Planning problem is devised. Next, the Line Planning model is integrated with the PESP model. The model is solved by decomposing the program back into the two sub-problems. Since the objective function consists only of variables related to the Line Planning component, the algorithm searches for optimal solutions for it and then tries to satisfy the PESP constraints.
3.5 Literature Gap

In the train planning literature, quality of service is typically considered in the line planning stage and modeled as the total number of *direct passengers*, as the total number of transfers or as the total ride time (on board time). Bussieck et al. (1996) suggests the minimization of *total travel time* of all passengers, including waiting time at origin and transfer stations as a better way to model quality of service. However, in their Line Planning model, as well as in all other studies in this domain, a timetable is not constructed, therefore it is impossible to calculate the *total travel time* of passengers while constructing the lines plan.

In timetabling models it is possible to measure the total travel time of passengers but the effect of decisions made at this stage on the total travel time is limited. Indeed, no study on train timetabling considered the total travel time as an objective function. Moreover, in many studies (e.g., Carey and Lockwood (1995), Higgins et al. (1997)) the objective function optimized at the timetabling stage is conflicting with the one that is typically considered in the previous planning stage. For example, minimizing cost vs. maximizing service.

We note that the optimal journey for each traveler is determined only when the timetable is known to the traveler. However, previous studies of Wong and Leung (2004) and Schöbel and Scholl (2005), assumed predetermined paths for each origin destination pair, independent of the availability of trains that directly connect them and on transfer times. This assumption greatly simplifies the traffic assignment problem but ignores the important component of the problem. In our study, the traffic assignment problem is solved based on the actual line plan and timetable. That is to say, passengers are assumed to plan their journey so as to minimize their arrival time at the destination, given their arrival time at the origin station.

Integration of the line planning and timetabling processes may allow improvement of the service provided by public transportation operators. However, the literature on integrated railway planning is sparse and limited. For example, Gorman (1998) avoids microscopic details related to safety and operational constraints. Lindner (2000) focus on cost objective function and hence does not take advantage of the integration.

In this study we show that integration of the line planning and timetabling problems allows the introduction of an appropriate objective function. The resulted integrated problem is harder to solve, but we devise an evolutionary algorithm that is capable of providing good line plans and timetables. Results for the instance of Israel Railways dominate the one that is used in practice and enable reducing the total travel time of the passengers by some 20%.
3.6 Cross Entropy

In this section we review the Cross Entropy (CE) meta-heuristic. This method was originally proposed by Rubinstein (1997) for solving rare-event simulation problems. Rubinstein (1999) extended the method for solving combinatorial optimization problems. The algorithm presented in this study uses the CE meta-heuristic to solve the integrated line planning and time tabling problem with some extensions presented in the sequel. A CE algorithm is an evolutionary algorithm that iteratively applies to the following two phases:

1. Generation of a sample random data according to a specified random mechanism.
2. Updating the parameters of the random mechanism, typically parameters of probability mass functions (PMF), on the basis of this data to produce a "better" sample in the next iteration.


3.6.1 Rare-Event Simulation

Let \( X \) be a random vector taking its value in some discrete space \( \chi \) with a PMF \( f(x) \), be a real-value function defined on \( \chi \) and \( \gamma \) be a real number. In the rare-event simulation context, one needs to estimate the probability of occurrence \( I \) of an event \( \{S'(x) \geq \gamma\} \). One way to estimate this probability is importance sampling. Through this process, a different PMF is generated in which the occurrence of the rare event is more likely. Given a different PMF \( g(x) \), a random sample \( X_1, X_2, ..., X_N \) is drawn and the probability of occurrence of the event is estimated through the following unbiased estimator

\[
\hat{l} = \frac{1}{N} \sum_{i=1}^{N} I_{\{S'(X_i) \geq \gamma\}} \frac{f(X_i)}{g(X_i)}
\]  

(1)

where \( I_{\{S'(X_i) \geq \gamma\}} \) equals 1 if \( S'(X_i) \geq \gamma \) and 0 otherwise.

The best way to estimate \( I \) is to adopt the 'ideal' importance sampling PMF

\[
g^*(x) = \frac{I_{\{S'(x) \geq \gamma\}} f(x)}{\hat{l}} \]  

(2)

Here lies the difficulty and in order to find the 'ideal' importance sampling PMF we need to know \( I \), but \( I \) is the unknown probability we originally want to estimate. The main idea of the CE method for rare event simulation is to find in an a priori set \( G \) of
PMF's defined on $\chi$, the element $g(x)$ such that its distance from the 'ideal' sampling distribution is minimal. A particularly convenient measure of distance between two PMF's defined on $\chi$ (e.g. $a(x)$ and $b(x)$) is the Kullback-Leibler distance

$$D(a, b) = \int a(x) \ln \frac{a(x)}{b(x)} \, dx \quad (3)$$

As aforesaid, we would like to find a PMF $g(x)$ that has minimal distance from the 'ideal' sampling distribution:

$$\argmin_{g \in \mathcal{G}} D(g^*, g) \quad (4)$$

The cross-entropy between two probability distributions is defined as:

$$H(a, b) = H(a) + D(a, b) = - \int a(x) \ln a(x) \, dx + \int a(x) \ln \frac{a(x)}{b(x)} \, dx = - \int a(x) \ln b(x) \, dx \quad (5)$$

where $H(a)$ is the entropy of $a(x)$ and $D(a, b)$ is the Kullback-Leibler distance between $a(x)$ and $b(x)$. Hence, (4) is mathematically equivalent to maximizing the cross-entropy on the set $\mathcal{G}$ of PMF's (here is where the method derives its name)

$$\argmax_{g \in \mathcal{G}} = \int f(x) I_{\{S^*(x) \geq \gamma\}} \ln g(x) \, dx \quad (6)$$

If the probability of occurrence $I$ is big enough, a good way to estimate the solution for (6) is to solve its stochastic counterpart, i.e. to draw a sample $X_1, X_2, \ldots, X_N$ according to $f(x)$ and to solve the following:

$$\argmax_{g \in \mathcal{G}} = \sum_{j=1}^{N} I_{\{S^*(X_j) \geq \gamma\}} \ln g(X_j) \, dx \quad (7)$$

3.6.2 Solving Combinatorial Problems with the CE Method

The ideas described in the previous section can be applied to solve combinatorial optimization problems. Assuming now that $S^*(x)$ is the maximal value of the function $S(x)$, that is $S^* = \max_{x \in \chi} S(x)$. The same concepts described above are used in order to find the values of $S(x)$, where $S(x) = S^*$, which are usually rare events. However, in combinatorial optimization problems, the value of $S^*$ is unknown (this is the value we are looking for). CE algorithms by-pass this by taking the following iterative scheme:

1. Draw a random sample $X_1, X_2, \ldots, X_N$ (Generation 1) according to a given PMF $g_1(x)$.
2. From this sample compute $\gamma_1 = \max_{x \in \{X_1, \ldots, X_N\}} S(x)$ and after that solve the stochastic counterpart given in (7), the solution is PMF $g_2(x)$.
3. Draw a new sample $X_1, X_2, \ldots, X_N$ (Generation 2) according to PMF $g_2(x)$.
4. From this compute $\gamma_2 = \max_{x \in \{X_1, \ldots, X_N\}} S(x)$ and after that solve the stochastic counterpart given in (7), the solution is PMF $g_3(x)$.
Continuing this way, a sequence of PMF’s $g_1(x), g_2(x), g_3(x)\ldots$ is computed. Assuming that the Generation size is large enough, as we proceed with this scheme, the new PMF’s are more likely to generate samples $X_1, X_2, \ldots, X_N$ with high values of $S(x)$.

If the probability of the rare event is extremely low, it can be assumed that only one observed element holds $S(X_i) = \gamma_t$ in each iteration $t$. Consequently a very large sample size will have to be used in order to make the stochastic counterpart accurate enough. To ensure that $g_t(x)$ is calculated based on enough elements, a different limit $\hat{\gamma}_t$ is used instead. The value of $\hat{\gamma}_t$ is set so that a fraction $\rho$ of the drawn elements (Elite) have a value $S(x)$ larger or equal to $\hat{\gamma}_t$.

$$\arg\max_{s \in G} = \sum_{j \in \text{Elite}} \ln g_j(x_j)$$  \hspace{2cm} (8)

**Smoothing** – to prevent premature convergence to a local maximum the PMF’s are updated in the following manner:

$$g_t(x) = \alpha h_t(x) + (1 - \alpha) g_{t-1}(x)$$  \hspace{2cm} (9)

where $h_t(x)$ is the solution for the stochastic counterpart in (8), as described above in the iterative scheme.

**CE algorithm for combinatorial optimization (maximization problem)**

<table>
<thead>
<tr>
<th>Parameters:</th>
<th>Generation size $G$, Elite size $F/S$, and a smoothing parameter $\alpha$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input:</td>
<td>An objective function $S(x)$, PMF $g_0(x)$.</td>
</tr>
<tr>
<td>Output:</td>
<td>A vector $x_{\text{max}}$ that approximately maximize $S(x)$ and its value.</td>
</tr>
<tr>
<td>Set:</td>
<td>$t = 0$ and $s_{\text{max}} = -\infty$</td>
</tr>
<tr>
<td>Repeat:</td>
<td></td>
</tr>
<tr>
<td>Step 1:</td>
<td>Draw a random sample $X_1, X_2, \ldots, X_{GS}$ from $g_t(x)$.</td>
</tr>
<tr>
<td>Step 2:</td>
<td>Find the Elite and set $s = \max_{x \in {x_1, \ldots, x_{GS}}} S(x)$</td>
</tr>
<tr>
<td>Step 3:</td>
<td>If $s_{\text{max}} &lt; s$ set $s_{\text{max}} = s$</td>
</tr>
<tr>
<td></td>
<td>and $x_{\text{max}} = \arg\max_{x \in {x_1, \ldots, x_{GS}}} S(x)$</td>
</tr>
<tr>
<td>Step 4:</td>
<td>Calculate PMF $g_{t+1}(x)$ via equation (9). Set $t = t + 1$.</td>
</tr>
<tr>
<td>Until:</td>
<td>Convergence is reached.</td>
</tr>
</tbody>
</table>

19
4 Problem Definition

In this chapter we introduce a model that combines the Line Planning Problem and the Train Timetabling Problem presented in Chapter 3. As mentioned before, in previous studies line planning and timetabling decisions are usually made in a hierarchic manner. In the formulation presented here, these decisions are made simultaneously.

The Integrated Line Planning and Timetabling Problem (ILPTP) is defined as follows: Given
- railway infrastructure description;
- a set of possible routes;
- cycle time and horizon time;
- passengers' demands for journeys (Origin-Destination Matrices);
- safety and operational restrictions;

Find a cyclic timetable that minimizes the total travel time of all passengers and satisfies the safety and operational restrictions. Decide on:
- the number of trains that serve each route in a cycle (route frequency);
- the stopping stations of each train;
- the times in the cycle in which each train enters and leaves its blocks and stations. (operational timetable)

Problem Parameters
The railway infrastructure is composed of a collection of blocks $B$, passenger stations $S$, operational stations $O$, and signals $N$. The infrastructure layout is represented by an undirected graph, $G = (S \cup O \cup N, B)$. Parallel arcs are allowed. Each block, $b \in B$ is characterized by its minimal traversing time, $TRAVELING\_TIME_b$. Each station node $s \in S \cup O$ is characterized by capacity $STATION\_CAPACITY_s$. Namely, the maximal number of trains that can occupy it simultaneously. This representation allows handling any type of railway network described in the literature and reviewed in Section 3.2.

![Figure 5: A graph representation of the network displayed in Figure 1 (page 3).](image)

A possible route can be any directed path on the abovementioned graph connecting two passenger station nodes. For example, in Figure 5 a possible route between stations 1001 and 1005 can be represented by the following sequence of nodes and arcs:


The number of possible routes grows exponentially in the number of blocks and stations. As discussed in the papers described in Section 3.1, it is common to define a
"Menu" of routes to choose from. The menu size can be significantly reduced by using considerations similar to those that train planners use manually, such as:

1. Selecting end point stations – a station can serve as an end point station of a route only if its capacity is big enough to allow parking of trains before or after a journey.
2. Defining traveling directions for each block in multi-block segments.

The OD-matrices represent the demand rates for journeys per time unit (e.g., minutes). Another two parameters of the planned timetable are CYCLE_TIME and HORIZON_TIME. As aforementioned, in a cyclic timetable all planned trains are repeated every cycle, CYCLE_TIME determines how many time units are in each cycle of the timetable (60 minutes in this study). HORIZON_TIME determines the length of the timetable (e.g., in our numerical study we consider a system that works 1140 minutes a day, which is 19 Cycles)

**Safety Restrictions**

In order to make the planned timetable feasible, the timetable has to comply with some safety restrictions:

1. Each block can be seized by only one train at a time.
2. Extra safety headway is required between two consecutive trains traveling on the same block in opposite directions (EXTRA_HEADWAY).

![Figure 6: Minimal headway](image)

In Figure 6, the trains are represented by black arrows. In situation A, the trains are traveling in the same direction, therefore the first restriction is assuring enough headway. In situation B, the trains are traveling in opposite directions. To avoid a possible conflict at the block edge, an extra time interval is required between the planned exiting time of train 1 and the planned entering time of train 2 to the block.

**Utilization Restriction**

BLOCK_UTILIZATION defines the total amount of time a block can be seized in a cycle. (e.g. Israel Railways planners allow 75% utilization of each block). In most cases a higher utilization rate of some of the blocks can be achieved without violating the safety restrictions. The utilization restriction exists in order to assure some robustness in the planned timetable. Forcing free time intervals on every block enables better recovery in occurrences of delays due to unpredicted events.

**Operational Restrictions**

A minimum dwelling time is required at the passenger stations in which the trains stop in order to allow embarking and disembarking of passengers (STOPPING_TIME). Some passengers may need to switch trains during their journey. TRANSFER_TIME is the minimal time required between the entering of a train and the departing of another train in order to allow such switching. This includes the time needed to change platforms and some extra time to manage occurrences of small delays.
The net traveling time of a train is defined as the sum of minimal times needed to traverse the blocks and minimal stopping times at passenger stations. A train may dwell an additional time at passenger stations or operational stations for safety and operational considerations. For example: waiting for the next block to become available or waiting for an over-taking train to pass. Consequently, the total traveling time of a train might be considerably longer than its net traveling time. MAX_DWELLING is the relative total extra time the train is allowed to dwell over all its stations. This restriction can be easily relaxed by setting MAX_DWELLING to a high value.

**Objective Function**
The model's objective is to minimize the total travel time of all passengers. This includes waiting time at origin station, time on board the trains and transfer times. In order to calculate the time a passenger will spend in the system, a number of assumptions are made on the way passenger traffic is assigned:

1. Each passenger is interested in getting to his destination as early as possible.
2. A passenger can embark on every train that stops in his station, namely, there is no capacity limit on the trains. We note that this is a standard assumption in the timetabling literature.

When a passenger arrives at his origin station he plans his trip so that he will arrive as soon as possible at his destination station. The time between the passenger’s arrival at the origin station and his arrival at the destination is referred to as the passenger's total travel time. The objective is to minimize the sum of the total travel times of all passengers.

**Penalties for unserved passengers**
Some of the demands may not be served by a planned timetable, for example:

- There is no optional trip between a pair of stations. For example, there is no planned stop at a certain station (either the origin or destination)
- A passenger arrives at the origin station near the end of the planning horizon and there are no more available trips to his destination in the rest of the horizon

For our objective function to be well defined, some penalty function should be used to account for unserved passengers. For example, a constant penalty can be associated with each unserved passenger or the penalty can be trip and time dependent. In the numerical experiment presented in this study, the objective is penalized for each unserved passenger by taking his travel time as the time between his arrival and the end of the planning horizon. The main flaw of this penalty is that it may be beneficial not to serve passengers that arrive near the end of the planning horizon. Typically these demands are significantly low and therefore will not affect the resulting cyclic timetable. In any case, this can be easily fixed by extending the planning horizon and adding some OD-matrices with zero demands. As aforesaid, we could use any other penalty function.
4.1 Sample problem

In this section an instance of the problem is presented together with a feasible solution. Although the presented instance is relatively small, the number of parameters that needs to be taken into account and the number of decision variables are considerably large. This may indicate on the complexity of real-world instances.

4.1.1 Input

The rail system in this instance has an infrastructure similar to the one presented throughout this study. In Figure 7, the numbers that appear above each segment represent the traveling times on the blocks of each segment (TRAVELING_TIME). The numbers that appear under the passenger stations represent the stations' capacities (STATION_CAPACITY).

![Figure 7: Sample problem - a graph representation of the network displayed in Figure 1, including minimal traveling times on the blocks and stations capacities.]

Menu of routes:


The maximal frequency allowed for each route is 4.

We refer to routes that start at station 1001 as outbound routes. Respectively, inbound routes are routes that end in station 1001.
OD-matrices:
4 identical matrices are given, one for each hour between 08:00-12:00 (Table 2). The demands are given in passengers per hour. In order to calculate the objective function, the demand is assumed to be uniformly distributed over the hour.

Table 2: Sample problem – origin-destination matrix

<table>
<thead>
<tr>
<th>D</th>
<th>O 1001</th>
<th>O 1002</th>
<th>O 1003</th>
<th>O 1004</th>
<th>O 1005</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>60</td>
<td>60</td>
<td>60</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>1002</td>
<td>60</td>
<td>60</td>
<td>60</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>1003</td>
<td>60</td>
<td>60</td>
<td>60</td>
<td></td>
<td>60</td>
</tr>
<tr>
<td>1004</td>
<td>60</td>
<td>60</td>
<td>60</td>
<td></td>
<td>60</td>
</tr>
<tr>
<td>1005</td>
<td>60</td>
<td>60</td>
<td>60</td>
<td></td>
<td>60</td>
</tr>
</tbody>
</table>

Additional parameters:

Table 3: Sample problem – additional problem parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CYCLE_TIME</td>
<td>60 min.</td>
</tr>
<tr>
<td>HORIZON_TIME</td>
<td>240 min.</td>
</tr>
<tr>
<td>EXTRA_HEADWAY</td>
<td>1 min.</td>
</tr>
<tr>
<td>BLOCK_UTILIZATION</td>
<td>75%</td>
</tr>
<tr>
<td>STOPPING_TIME</td>
<td>2 min.</td>
</tr>
<tr>
<td>TRANSFER_TIME</td>
<td>4 min.</td>
</tr>
<tr>
<td>MAX_DWELLING</td>
<td>30%</td>
</tr>
</tbody>
</table>
4.1.2 Solution

A feasible solution for the above instance was constructed and it is not necessarily optimal. The Time-Distance diagram in Figure 8 displays the planned train trips for one hour. The data in this diagram is equivalent to an operational timetable. The planned timetable is cyclic, therefore it is enough to check it's feasibly in a single hour. The resulting timetable will consist of two different lines: line 1001-1004 and line 1001-1005. In order to present the trains of both lines on the same diagram, line 1001-1004 is divided into two parts. The slots of the trains between station 1001 and 1002 are presented with the trains of line 1001-1005. The trains' continuations, between stations 1002 and 1004, are presented above the separating gray area. The continuations of the trains of line 1001-1004 are connected with dotted lines. The resulting timetable will consist of two different lines: line 1001-1004 and line 1001-1005. In order to present the trains of both lines on the same diagram, line 1001-1004 is divided into two parts. The slots of the trains between station 1001 and 1002 are presented with the trains of line 1001-1005. The trains' continuations, between stations 1002 and 1004, are presented above the separating gray area. The continuations of the trains of line 1001-1004 are connected with dotted lines. The red lines in the diagram represent inbound trains and the blue lines represent outbound trains. The segments between stations 1002 and 1005 are built of two unidirectional blocks, lines with different colors can cross each other since the trains are using different blocks.

Figure 8: A solution for sample problem – Time-Distance diagram

Figure 9 illustrates the planned seizing of the bi-directional segments connecting stations 1001 and 1002. The seizing of these blocks comply with all the relevant safety and operational restrictions:

- There are no overlapping slots and in addition, a minimal gap is kept between every red-blue pairs of slots (representing trains traveling in opposite directions).
- Block 2001 is seized in total for 42 minutes; Block 2002 is seized in total for 30 minutes. The maximal time allowed is 45 minutes - 75% of the cycle time.

![Table showing the seizing of blocks between stations 1001 and 1002]

Figure 9: A solution for sample problem – blocks display
The data in Table 4 represents the timetable that will be published to the passengers. If a train does not stop in a certain station, vertical dashed line appears in the relevant cell. The timetable is built of 3 inbound trains and 3 outbound trains that are repeated every hour. For example trains 109,209,309 are replications of train 9 obtained by shifting their times by one, two and three cycles (hours). Trains that finish their planned journeys after the end of the planning horizon (12:00) are omitted (train 311).

Example for train’s total and net travel times calculation: Train 4 is traveling in route (2005,2002,2001); the sum of travel times on these blocks is 18 minutes. In addition, the train stops at station 1002, the minimal stopping time is 2 minutes, therefore the net traveling time of the train is 20 minutes. The train begins its journey at 08:23 end finishes it at 08:48, that is, the total travel time of the train is 25 minutes. The extra dwelling time is 5 minutes which is 25% of the net traveling time. This complies with the 30% MAX_DWELLING restriction.

Next we demonstrate the calculation of passenger total travel time. For example, a passenger arrives in station 1004 at 08:50 and is interested in traveling to station 1005. The earliest time the passenger will be able to reach station 1005 is 09:50, according to the following trip plan:
- Waiting in station 1004;
- Embarking on train 104 at 09:23;
- Disembarking train 104 in station 1002 at 09:29
- Waiting in station 1002;
- Embarking on train 109 at 09:33;
- Arriving in station 1005 at 09:50.

The total travel time of the passenger is 60 minutes.

Summing up the total travel time of all passenger flows according to this approach yields a sum of 161,069 minutes, which is the value of this solution. There is a demand for 4800 journeys in total; therefore the average time a passenger spends in the system is 161,069/4800=33.55 minutes.
5 Methodology

In this chapter a heuristic method for solving ILPTP will be presented. Integration of the Line Planning Problem and the Train Timetabling Problem is not common in the literature. In most cases, when researchers try to solve real-world instances of one of these problems they eventually turn to applying heuristic methods. In addition, in some cases researchers are forced to make some simplifying assumptions. Sometimes these simplifying assumptions make the resulting solution impractical. Integrating the two problems creates a more difficult problem. In particular, the problem that was formulated in the previous chapter is even more difficult to deal with. The reason for this is because in order to evaluate a solution, the journey characteristics of each flow of passengers have to be determined.

In this chapter we present an insertion method that can quickly create a large set of feasible solutions for the ILPTP. Then an algorithm that evaluates the objective function of each such solution is introduced and these two algorithms are used as subroutines for the CE algorithm. The CE algorithm requires an encoding scheme of the solutions. A good encoding method should be compact and meaningful for the CE update mechanism.

The evaluation of a timetable is carried out by assigning all flows of passengers to specific journeys. It is assumed that each passenger selects a journey that minimizes his arrival time at the destination. Determining these journeys is equivalent to finding a shortest path on a special graph that represents the timetable. We refer to this graph as the events graph. Calculation of all pairs of shortest paths of this graph can be done using known algorithms such as the Floyd-Warshall algorithm and Johnson's algorithm (see e.g., Cormen et al. (2001)), but this may require long computation time. In Section 5.4 we will show how we can take advantage of the properties of the events graph in order speed up this computation.

The rest of this chapter is organized as follows: First a method for encoding a feasible solution is presented. Afterwards methods for decoding a timetable and evaluating it are introduced. Finally an implementation of the CE method on the solution coding is presented.
5.1 Solution Encoding

As described in the previous chapter, in order to construct line plans and a timetable, the following decisions need to be made:

- the number of trains that serve each route in a cycle (route frequency); a possible value for this parameter is zero, namely, the line is not in use.
- the stopping stations of each train;
- the times in the cycle in which each train enters and leaves its blocks and stations (operational timetable).

The model's input consists of the maximal frequency of each route. It may be determined by the planner (based on considerations that are not included in the model) or may be calculated by the following upper bound:

\[
\left( \frac{CYCLE\_TIME \cdot BLOCK\_UTILIZATION}{\max_{(all\ blocks\ in\ route)} TRAVELLING\_TIME_b} \right)
\]

The maximal frequency of the route determines the maximal number of trains that may be inserted in this route during a cycle (e.g., 60 minutes). We refer to each one of these trains as a possible train. Each possible instance of a train within a cycle is represented by a *gene* containing the following decisions variables:

- IN_USE - Boolean stating whether or not this train is being used.
- EARLIEST_TIME - The earliest time (in a cycle) that the train can be inserted at the first block.
- STOPPING_STATIONS – Set of passenger stations where the train stops. This set is represented by a characteristic (Boolean) vector.

![Figure 10: a gene](image-url)
A feasible solution is represented by a set of genes, one for each possible train instance. Such a set is referred to as a chromosome. Each gene is related to a given route and the number of genes related to each route is determined by the maximal frequency of the route.

The frequency of each route will be determined by the number of possible trains that will eventually be scheduled by the algorithm. Some blocks may be used by more than one route. Therefore, it is often the case that not all routes can be scheduled to their maximal frequencies.

Note that the chromosome encodes the earliest time in a cycle a train can be inserted in the first block of its route. The actual arrival and departure times at each block depends on the availability of the train and the blocks. Therefore, after inserting the train to its first block, the search space is significantly reduced. In the next section, we present a method to determine the arrival and departure times at each block.

5.2 Decoding one cycle

This work focuses on creating cyclic timetables. In this section a routine for inserting trains into one cycle will be presented, i.e. determining all arrival/departure times in terms of one cycle. In the following section it will be explained how this cycle is being repeated over the planning horizon to create a full cyclic timetable.

Each route in the menu is given a priority. A train that travels on a route with a higher priority can bypass low priority trains. For example: intercity trains can bypass regional trains. Trains are inserted in decreasing priority order, hence when inserting each train, one must comply with scheduling decisions related to all trains with higher priorities. In this study it is assumed that priorities can be predetermined by the planners, this allows introducing considerations that are not formally included in model. This assumption agrees with the practice of some railway companies (e.g., the Israeli Railway). Alternatively, the priorities could be variables whose optimal values could be searched through the CE algorithm. In addition, each route's traveling direction is marked as inbound or outbound. All possible outbound trains will be

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>1011</td>
</tr>
<tr>
<td>0</td>
<td>38</td>
<td>1001</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>101110101</td>
</tr>
<tr>
<td>1</td>
<td>46</td>
<td>111000111</td>
</tr>
<tr>
<td>0</td>
<td>11</td>
<td>1000011</td>
</tr>
<tr>
<td>1</td>
<td>42</td>
<td>1111111</td>
</tr>
<tr>
<td>1</td>
<td>30</td>
<td>111100011001</td>
</tr>
<tr>
<td>0</td>
<td>58</td>
<td>100001111011</td>
</tr>
</tbody>
</table>

Figure 11: A part of a chromosome
given odd numbers and all possible inbound trains will be given even numbers (similar to what Israel Railway planners do). Later, in the insertion process, the trains' numbers will help in identifying the traveling direction of the trains inserted on each block.

Creating a timetable based on chromosome (decoding) is done as follows: first, all genes with True values of IN_USE are sorted by their priorities (ties are broken by EARLIEST_TIME value). Later, the algorithm tries to schedule these possible trains one by one, similarly to the manual process done by planners. A possible train is said to be scheduled if the algorithm was able to insert it in all the blocks of its route and if the total journey time does not exceed the MAX_DWELLING restriction. The slot times fully determine the stopping times in the stations. The time gap between consecutive slots of a train is the time the train spends in the connecting station. After each insertion of two consecutive slots, the algorithm checks that the STATION_CAPACITY of the connecting station is not being violated. If this restriction is violated, the train can not be scheduled. The exiting time from one block will be used as the earliest time to insert in the next block. In cases where the STOPPING_STATION variable related to the connecting passenger station has a True value, the earliest time to insert in the next block will be advanced by STOPPING_TIME.

5.2.1 Insertion Process - Block Seizing and Station Seizing

Each block has CYCLE_TIME time units that can be seized by trains. The algorithm keeps track of the occupancy in each block. The total block occupancy may not exceed CYCLE_TIME * BLOCK UTILIZATION. Each time unit can be seized by only one train. In order to insert a train to a block, the algorithm looks for a TRAVELLING_TIME sequence of time units that are not seized, such that the headway constraint is satisfied. The search starts at the given earliest time.

For each station and siding the algorithm keeps track of the number of trains that occupy it in every time unit. The algorithm does not choose the platforms for the trains in each station (Platforming), instead it verifies that the station capacity restriction holds. If a train enters a station at the same moment another one leaves, the station occupancy will not change, but this may lead to a collision. To avoid these situations, a train will occupy a station one time unit before it enters the station and one time unit after it leaves. A train is scheduled if it can be inserted in all the blocks and stations along its route without violating the block utilization and max dwelling constraints. The insertion procedure is formally described by the pseudo code below.
Pseudo code for Insertion Process:

Create a list of all genes having True value of IN_USE sorted by their route priorities
For all genes g in the list
{
    Set TIME to g.EARLIEST_TIME;
    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 list
    {
        Find the earliest time sequence of time units that start later than Time (modulo CYCLE_TIME) in which the train g can be inserted
        If unable to insert then break; (this train can not be scheduled)
        If current block follows a station or a siding
        {
            Find the time units in which the train dwells at the station/siding;
            Enlarge occupancy of the station/siding at these time units;
            Check station/siding capacity;
            If Capacity is violated then break; (this train can not 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 occupancy of the previous block then
                    extend the train's previous slot;
                else
                    break; (this train cannot be scheduled)
            }
            If this block is followed by a station or a siding then
                Try to shorten and shift all previous slots forward in the sequence;
        }
        Set TIME to exit time from current block
        If this block is followed by a station with True value of STOPPING_STATIONS
            Set Time to (Time+STOPPING_TIME) modulo CYCLE_TIME;
    }
    Check capacity in last station;
    If Capacity is violated, this train can not be scheduled;
    Calculate train net travelling time;
    Calculate train total travel time;
    If total travel time exceeds MAX_DWELLING constraint, this train can not be scheduled;
    Repeat for all stations in route
    {
        If dwelling time in station is bigger or equal than STOPPING_TIME and the station has a False value of STOPPING_STATIONS, set STOPPING_STATIONS to True;
    }
5.2.2 Sample Problem

The timetable that was presented in the previous chapter was constructed according to the following chromosome:

<table>
<thead>
<tr>
<th>Route Number</th>
<th>Train Number</th>
<th>IN_USE</th>
<th>EARLIEST TIME</th>
<th>STOPPING_STATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>9</td>
<td>1 1 1</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1</td>
<td>18</td>
<td>1 1 1</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>1</td>
<td>41</td>
<td>1 1 1</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>0</td>
<td>45</td>
<td>1 0 1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>8</td>
<td>1 1 1</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1</td>
<td>23</td>
<td>1 1 1</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>1</td>
<td>36</td>
<td>1 1 1</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0</td>
<td>51</td>
<td>1 1 1</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>1</td>
<td>17</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>1</td>
<td>27</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>1</td>
<td>44</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>1</td>
<td>51</td>
<td>1 1 0 1</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>1</td>
<td>26</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>0</td>
<td>44</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>0</td>
<td>57</td>
<td>1 0 1</td>
</tr>
</tbody>
</table>

First, all genes having true value in IN_USE are sorted. In this example all 4 routes have the same priority, therefore the genes will be sorted in ascending order of EARLIEST_TIME. The sorted list of genes is:

<table>
<thead>
<tr>
<th>Route Number</th>
<th>Train Number</th>
<th>IN_USE</th>
<th>EARLIEST TIME</th>
<th>STOPPING_STATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>8</td>
<td>1 1 1</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>1</td>
<td>17</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>18</td>
<td>1 1 1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>1</td>
<td>23</td>
<td>1 1 1</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>1</td>
<td>26</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>1</td>
<td>27</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>1</td>
<td>36</td>
<td>1 1 1</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
<td>41</td>
<td>1 1 1</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
<td>1</td>
<td>44</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>1</td>
<td>51</td>
<td>1 1 0 1</td>
</tr>
</tbody>
</table>
When trying to insert the first train (10), all blocks are empty. The train can be inserted at EARLIEST_TIME to the first block in its route. According to the gene there are planned stops at stations (1002, 1003). As can be seen in Figure 12, there is a time gap between the slots in blocks 2004 and 2008 and between the slots in blocks 2004 and 2002. Naturally, since this is the first train to be inserted, its total travel time equals to its net traveling time.

The next train to be inserted is train 2. The train is inserted to block 2006 between the times 8-14. The train has to stop for at least 2 minutes in station 1002, the earliest time to insert to block 2002 is 16, but the train cannot enter block 2002 before the 25th minute. The train will be inserted in block 2002 between the times 25-30 and in block 2001 between the times 32-39. The total travel time is 31 minutes, the net traveling time is 20 minute. This does not comply with the MAX_DWELLING restriction. The train is not scheduled.

Train 9 can be scheduled. Between the times 24-26 the train is dwelling at siding 3001, allowing train 10 to pass.
The EARLIEST TIME to insert train 3 to block 2001 is 18, given the already scheduled trains, the earliest possible time to insert the train to this block is 33.

Inserting train 4:
The train will dwell in siding 3001 between the times 35-41.

Inserting Train12:
Inserting train 11: the slot in block 2001 demonstrates the cyclicity principle. The train is inserted between the times 59-6.

At this point, block 2001 is occupied for a total time of 42 minutes. It is not possible to insert more trains to this block. Since all other trains in the list travel through this block, the algorithm will not be able to schedule anymore trains. Figure 17 represents one cycle of the timetable at the end of the insertion process.

5.2.3 Sequence of Blocks

We refer to a set of several consecutive blocks separated only by signals (rather than stations) as a sequence of blocks. While traveling on a sequence of blocks, the exiting time from one block must be the entering time to the next. Note that if two blocks are separated by a station, the exiting time from the first may occur arbitrarily earlier than the entering time to the second.

Consider such a sequence of two blocks separated by a signal: block A followed by block B. If block B cannot be seized immediately after block A, the insertion algorithm performs the following operations:

- Look for the earliest time to insert the train in block B and insert it.
- Check whether it is possible to extend the slot in block A until the entering time to block B (i.e., slowing down the train).
- If the extension of the slot is not possible, the train can not be scheduled.
- If the extension of the slot is possible, the train can be inserted to these blocks. The extended slot represents traveling in lower speed on block A.
5.2.4 Traversing Blocks in Minimal Time

Trains with low priority will sometimes need to dwell an additional time in stations or travel at low speeds. There are two main reasons for preferring additional dwelling at a station over traveling at lower speeds:

1. As resources, blocks are usually more expensive than stations. It is generally easier to add a platform in a station rather than build a new track section. If a train has to delay, it is better to seize a platform than a block. A train that dwells on a block creates a "traffic jam" for all trains traveling through this block.

2. Dwelling in passengers' stations is always better from the objective function perspective. See Proposition 2, below.

Under the three assumptions listed below we can show that in an optimal solution it is always possible to schedule all the trains in their maximal speed.

Assumption 1: The minimal traveling time on a block is equal for all train types.

Assumption 2: The station capacity constraint is not binding.

Assumption 3: All the railway junctions are located at stations (either passenger or operational stations). In Figure 18, A satisfies this assumption but B does not.

![Figure 18: Infrastructure parts that satisfy (A) and do not satisfy (B) Assumption 3.](image)

**Proposition 1:** Under the assumptions above, any feasible schedule can be modified such that all trains run at their maximal speed. Furthermore, for any sequence traversed during the time interval \([e, d]\) in the original solution and during \([e', d']\) in the modified solution, \([e', d'] \subseteq [e, d]\).

**Proof:**

We consider two distinct cases:

- Traveling on a single block that connects two stations.
- Traveling on a sequence of blocks.

The first case is trivial, if a train is scheduled to traverse the block slower than the minimal traveling time, then we can delay the train at the previous station and seize the block for a shorter time without creating any conflicts with other trains. Indeed the train may seize resources in the origin station for a longer period, but under Assumption 2 above, this may not affect the feasibility of the new solution.

For the case of a sequence of blocks, assume that the trains traverse between two stations during the time interval \([e, d]\). First note that under Assumption 3 above, trains traveling in opposite directions cannot simultaneously seize blocks in the sequence. In addition, trains that travel in the same direction cannot overtake each
other within the sequence. Explicitly, under Assumption 3 each block in the sequence is traversed by the same set of trains and in the same order.

We denote the entering time of train $i$ to block $j$ by $t_{ij}$ and the traveling time of train $i$ on block $j$ by $p_{ij}$. Recall that a feasible schedule is continuous, i.e. $t_{ij} + p_{ij} = t_{i,j+1}$ for all $i$ and $j$. In a sequence of $n$ blocks, $t_{i,n+1}$ represents the exit time from the last block in the sequence.

Given Assumption 1, we denote the minimal traveling time on block $j$ as $\text{traveling\_time}_j$. Block $k$ is one that maximizes $\text{traveling\_time}_i$, that is

\[ \text{traveling\_time}_k \geq \text{traveling\_time}_j \quad \forall j \]  

We first show that the entering times to block $k$ and the entering and exiting times to all blocks prior to it can be postponed such that these blocks are traversed in minimal times. This is done without modifying the exit times from block $k$ (entering time to block $k+1$). All entering times to blocks prior to the $k+1$ block are set to:

\[ t'_{ij} = t_{i,k+1} - \sum_{l=i}^{l=k} \text{traveling\_time}_l \]  

Since the original schedule is feasible, then, on block $k$ there is at least $\text{traveling\_time}_k$ time gap between the exiting times of every pair of consecutive trains $(i,i+1)$, that is,

\[ t_{i+1,k+1} \geq t_{i,k+1} + \text{traveling\_time}_k \]  

We will now show that in the modified schedule there is no overlap between any pair of consecutive trains on any block prior to block $k$. Namely, for any pair $t'_{i,j}, t'_{i+1,j}$:

\[ t'_{i+1,j} \geq t'_{i,j} + \text{traveling\_time}_j \]  

Subtracting $\sum_{l=j}^{l=k} \text{traveling\_time}_l$ from both sides of (13):

\[ t_{i+1,k+1} - \sum_{l=j}^{l=k} \text{traveling\_time}_l \geq t_{i,k+1} - \sum_{l=j}^{l=k} \text{traveling\_time}_l + \text{traveling\_time}_k \]

\[ t'_{i+1,j} \geq t'_{i,j} + \text{traveling\_time}_k \]  

Since (11) and (16) hold, (14) is true for any pair of consecutive trains on any block prior to block $k+1$. Therefore, the modified schedule is feasible. The last logical step is valid because all the blocks are traversed by the trains in the same order (Assumption 3) and since the minimal traveling time on each block is equal for all trains (Assumption 1).

In the same manner, all entering and exiting times from blocks following block $k$ until the end of the sequence can be set to earlier times such that these blocks are traversed in minimal times. All entering times to blocks following block $k+1$ are set to:
It can be shown that the modified schedule is feasible using the similar argument.

For each train $i$ the original traversing time interval on the sequence of blocks is denoted by $[e_i, d_i]$. This interval can be calculated by the following equations:

$$ e_i = t_{i,k+1} - \sum_{l=1}^{l=k} p_l \quad (18) $$

$$ d_i = t_{i,k+1} + \sum_{l=k+1} p_l \quad (19) $$

The modified time intervals of the trains can be calculated by the following equations:

$$ e_i' = t_{i,k+1} - \sum_{l=1}^{l=k} \text{travelling time}_l \quad (20) $$

$$ d_i' = t_{i,k+1} + \sum_{l=k+1} \text{travelling time}_l \quad (21) $$

In the original schedule some trains may not traverse all blocks in minimal times, that is any train $i$ and any block $j$: $p_{ij} \geq \text{travelling time}_j$. Hence, for any train $i$ $e_i' \geq e_i$ and $d_i' \leq d_i$. Q.E.D

This setting is similar to a no-wait flow-shop model where prolonging of processes is allowed, see for example Abadi et al. (2000). Analogously, in cases where all jobs are identical, any feasible solution can be modified to a feasible solution for the no wait flow-shop where prolonging is not allowed.

**Proposition 2:**
Under assumption 1-3 above, there exists an optimal solution in which the traveling time of all trains on all blocks is minimal.

**Proof:** By Proposition 1 we know that it is always possible to modify any solution to a feasible solution in which all blocks are traversed in minimal time by moving the departure time at the origin station forward and moving the arrival time at the destination station backward. Consider two stations A and B connected by a sequence of blocks and denote the original (resp., modified) departure time from A by $e$ ($e'$). Recall that the objective function to be minimized is the total travel time of all passengers. Clearly, the total travel time of passengers on a train traveling from A to B that boarded at a station other than A is not affected by increasing $e$. Passengers that boarded the train at station A and arrived before time $e$ are also not affected. Finally, passengers that arrived at station A during the interval $(e,e')$ are better off now because they may board an earlier train. Similarly, denote the original (resp., modified) arrival time to station B by $d$ ($d'$). Decreasing $d$ to $d'$, decreases the total
travel time of passengers traveling on the train whose destination is station B. In
addition, some other passengers may be able to catch earlier connections in station B,
thus also benefiting from the modification. Since the following departure/arrival times
of the train are left unchanged, other passengers will not be affected from this
modification. Q.E.D

Note that a timetable where trains are scheduled to traverse the blocks in minimal
time may allow the scheduling of additional trains. Adding a train to a given timetable
can only reduce the total travel time of all passengers.

The lesson from the above two propositions is implemented in the insertion algorithm.
After inserting a train to a sequence of blocks, the algorithm will try to shorten and
shift slots in sequences that consist of some extended slots.

5.3 Constructing a Timetable

At this stage we hold all planned arrival/departure times modulo CYLCE_TIME. In
order to build an operational timetable one has to duplicate the planned cycle over the
planning horizon.

Since we are evaluating the timetable from the passengers' perspective, we only need
to know the arrival and departure times at passengers' stations (published timetable).
The published timetable may be viewed as a list of arrival/departure events at
passenger stations. Each event is a tuple that contains the following information:

- Station Number
- Train Number
- Time
- Type (Arrival/Departure)

The events list is built in two steps:

- Spreading the trains that start in the first planned cycle. Note that the journey
times of some of these trains may exceed a cycle. Whenever a cycle is exceeded a
CYCLE_TIME is added to time of the following events.
- Duplicating the spread trains over the planning horizon, but excluding trains that
may exceed it.
Pseudo code for creation of events list:

For each scheduled train T
{
    Let block_list be the sequence of all blocks of the route of T
    Create departure event in the origin station at the entering time to the first block
    Set TIME to entering time to the first block
    Set NUM to 0
    For each block b in the list
    {
        Set ENTER to the exit time from the block
        TIME = TIME + (ENTER-TIME) mod CYCLE_TIME;
        If b is not the last block in the list
        {
            Set EXIT equal the entering time to the following block;
            TIME = TIME + (EXIT-ENTER) mod CYCLE_TIME;
            If b is followed by a passenger station
            {
                If (EXIT-ENTER) mod CYCLE_TIME >= STOPPING_TIME
                {
                    Create arrival event E in this station
                    E.time = TIME - (EXIT-ENTER) mod CYCLE_TIME;
                    Create departure event F in this station at F.time = TIME;
                }
            }
        }
        Else
        {
            Create arrival event in the last station at TIME;
        }
    }
    For I = 1 to \(\lceil HORIZON\_TIME \div TIME \rceil \div CYCLE\_TIME \rceil 
    {
        For each event E of the train T
        {
            Create a new event F;
            Copy the details of E to F;
            Set F.time to E.time + I*CYCLE\_TIME;
            Set F.train number to E.train number + I*1000
        }
    }
}
Return list of events;

For example: we look at the arrival/departures times of train 11 of the above planned cycle (Section 5.2.2). The train is scheduled to seize the first block from minute 59 to minute 6, the second at 6-11, the third at 13-17, the fourth 19-24 and the fifth at 24-30. Events list of train 11:

<table>
<thead>
<tr>
<th>Station</th>
<th>Train</th>
<th>Time</th>
<th>Event type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>11</td>
<td>59 (08:59)</td>
<td>2</td>
</tr>
<tr>
<td>1002</td>
<td>11</td>
<td>71 (09:11)</td>
<td>1</td>
</tr>
<tr>
<td>1002</td>
<td>11</td>
<td>73 (09:13)</td>
<td>2</td>
</tr>
<tr>
<td>1003</td>
<td>11</td>
<td>77 (09:17)</td>
<td>1</td>
</tr>
<tr>
<td>1003</td>
<td>11</td>
<td>79 (09:19)</td>
<td>2</td>
</tr>
<tr>
<td>1005</td>
<td>11</td>
<td>84 (09:24)</td>
<td>1</td>
</tr>
</tbody>
</table>
The train is spread on two cycles and in this example there are four cycles in the planning horizon, therefore the train can be duplicated twice.

Events list of duplicates:

<table>
<thead>
<tr>
<th>Station</th>
<th>Train</th>
<th>Time</th>
<th>Event type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>111</td>
<td>119 (09:59)</td>
<td>2</td>
</tr>
<tr>
<td>1002</td>
<td>111</td>
<td>131 (10:11)</td>
<td>1</td>
</tr>
<tr>
<td>1002</td>
<td>111</td>
<td>133 (10:13)</td>
<td>2</td>
</tr>
<tr>
<td>1003</td>
<td>111</td>
<td>137 (10:17)</td>
<td>1</td>
</tr>
<tr>
<td>1003</td>
<td>111</td>
<td>139 (10:19)</td>
<td>2</td>
</tr>
<tr>
<td>1005</td>
<td>111</td>
<td>144 (10:24)</td>
<td>1</td>
</tr>
<tr>
<td>1001</td>
<td>211</td>
<td>179 (10:59)</td>
<td>2</td>
</tr>
<tr>
<td>1002</td>
<td>211</td>
<td>191 (11:11)</td>
<td>1</td>
</tr>
<tr>
<td>1002</td>
<td>211</td>
<td>193 (11:13)</td>
<td>2</td>
</tr>
<tr>
<td>1003</td>
<td>211</td>
<td>197 (11:17)</td>
<td>1</td>
</tr>
<tr>
<td>1003</td>
<td>211</td>
<td>199 (11:19)</td>
<td>2</td>
</tr>
<tr>
<td>1005</td>
<td>211</td>
<td>204 (11:24)</td>
<td>1</td>
</tr>
</tbody>
</table>

Repeating this process for all trains in the planned cycle creates the events of all the trains in the published timetable. The events list consists of all these events. In the next section we will explain how this list is used for the evaluation of the timetable.
5.4 Timetable Evaluation

The objective function of the ILPTP is to minimize the sum of total travel times of all passengers. A journey is defined as a sequence of embarking stations, trains and disembarking stations used by a passenger to reach from her/his origin station to her/his destination. In order to evaluate a given timetable (events list), we first need to assign journeys to passengers. We assume that each passenger chooses a journey that allows arrival at the destination as early as possible. A passenger flow is characterized by origin, destination and arrival time at origin. We assign each passenger flow to a journey that minimizes its total travel time.

Next, an algorithm for finding each passenger flow’s journey and calculating its total travel time is presented. This algorithm is presented as part of the solution method for ILPTP, but can actually be viewed as a component of a Decision Support System where a timetable is constructed manually. At the construction process, the expected total travel time of all passengers can be quickly evaluated, allowing fast examination of different alternatives.

The list of events can be viewed as a directed graph where events are represented by nodes. Each pair of consecutive events of a train or of a station is connected with an arc. This graph is similar to the time-distance diagram presented above, with two differences:

- There are additional arcs connecting consecutive events of the same station, the green arcs in Figure 19.
- The diagram resolution is the passengers' stations. The entering/exiting time to each block or siding is not relevant for the passengers or for the timetable evaluation.

\[
\text{Figure 19: A solution for sample problem – a part of the events graph/}
\]
In this form not all paths on the graph represent feasible passenger journeys. If a train leaves a station a moment after another train is entering it, the trains will be connected by an arc. But a passenger would not be able to switch from the entering train to the exiting train. For example, in Figure 19, a passenger traveling on the blue train and arriving to station 1002 at 09:31 will not be able to embark on the red train at 09:32. A passenger arriving on the red train to station 1002 at 09:29 will be able to embark on the blue train at 09:33.

We are interested in creating a graph where all paths represent feasible passengers' journeys, namely to incorporate the TRANSFER_TIME constraint. The current graph is used to create a new graph (same nodes but different arcs), using the following rules:

Each arrival event will be connected to the following events:

- The next arrival event of the same train – arriving at the next station (represents staying on the train).
- The first departure event from the station that is at least TRANSFER_TIME after the event's time (represents changing platforms in the station and waiting for another train).

Each departure event will be connected to the following events:

- The next arrival event of the same train – arriving at the next station.
- The next departure event in the station (represents waiting at the station for the next train).

Calculating the earliest time a passenger can reach his destination is equivalent to finding the shortest path from the following event in the origin station to an event 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, we can take advantage of two special properties of the problem in order to devise a more efficient algorithm for finding the optimal journeys. First, the events graph has a special structure, it is directed acyclic with maximal out degree of two. Second, we are only interested in finding a path to the earliest node of each station.

5.4.1 Reachability Algorithm

Reachability time is referred to the earliest time that a station can be reached from a given event. For each event $E$ we define a vector that stores the reachability time to each station. The main idea of our reachability algorithm is based on the fact that the reachability time from an event is the minimal between the reachability times from the two following events. In order to calculate these reachability times, we loop through the events in a non increasing order of time. Below we present a pseudo code of the reachability algorithm. The program assumes the original event list as an input without the modification due to transfer times as discussed above.
Pseudo code for the reachability algorithm:

```
Sort events list in non increasing order of time;
For each event E in the event list
{
    If the type of E is arrival
    {
        Set NEXT_ARRIVAL to the next arrival event of the same train;
        Set NEXT_DEPARTURE to the next departure event in the same station that is at least
        TRANSFER_TIME later;
    }
    Else
    {
        Set NEXT_ARRIVAL the next arrival event of the same train;
        Set NEXT_DEPARTURE to the next departure event in the same station;
    }
    For stations S
    {
        If S is the station of event E
            Set E.reachability_time(S) = E.time
        Else
            {
                If NEXT_ARRIVAL is not NULL and NEXT_DEPARTURE is not NULL
                    E.reachability_time(S) = min(NEXT_ARRIVAL.reachability_time(S),
                    NEXT_DEPARTURE.reachability_time(S));
                Else If NEXT_ARRIVAL is NULL and NEXT_DEPARTURE is not NULL
                    E.reachability_time(S) = NEXT_DEPARTURE.reachability_time(S);
                Else If NEXT_ARRIVAL is not NULL and NEXT_DEPARTURE is NULL
                    E.reachability_time(S) = NEXT_ARRIVAL.reachability_time(S);
                Else
                    Set E.reachability_time(S) = \infty
            }
    }
    Sort events list in ascending order of time;
    Return list of events;
```

Alternatively, the reachability algorithm can be formulated as the following dynamic program. Here it is assumed that the modification of the graph to accommodate the transfer times has already been done.

**Reachability algorithm – Dynamic programming formulation**

- $F.time$ - Time of event $E$.
- $F.station$ - The station of event $E$.
- $\sigma_E$ - The following event in station $E.station$.
- $\rho_E$ - The following event of train $E.train$.

$Reach(E, S)$ - Earliest time to reach station $S$ from event $E$.

Bellman Equation:

$$Reach(E, S) = \min\{Reach(\sigma_E, S), Reach(\rho_E, S)\}$$

Proceed through the list of events in non-increasing order of time.
If $S = E.station$ then $Reach(E, S) = E.time$.
If $\sigma_E$ is undefined then $Reach(\sigma_E, S) = \infty$, e.g., this is the last event in the station.
If $\rho_E$ is undefined then $Reach(\rho_E, S) = \infty$, e.g., this is the last event of the train.
The reachability algorithm has two main steps:

- Sorting the events in a non-increasing order. The complexity of this operation is $O(N \log N)$ where $N$ is the number of events.
- Calculating the reachability times for each event (the main loop). The complexity of this phase $O(N \cdot K)$ where $K$ is the number of passenger stations.

Typically $\log N < K$, and so the complexity of the algorithm is determined by the main loop. In this case the complexity of the algorithm is $O(N \cdot K)$ which is linear in the output size. Namely, the algorithm admits the minimal possible complexity. In the unlikely situation where $\log N > K$, the complexity of the algorithm is $O(N \log N)$.

In our quick and dirty C implementation of the algorithm, instances with some 10000 events and 47 stations can be solved in less than 0.07 seconds on a modest desktop computer.

### 5.4.2 Calculation of the Objective Function Value

Having the reachability times from each event, for each flow of passengers we look for the earliest departure event in the origin station. The difference between the reachability time to the destination in this event and the arrival time to the origin station is the total travel time of this flow of passengers. For each flow we calculate the total travel time and multiply it by the number of passengers in the flow. The sum of these calculations over all passengers’ flows is our objective function value.

![Figure 20: A passenger planning her journey](image)
5.5 Cross Entropy Application

In the previous sections, methods for fast creation and evaluation of cyclic timetables were introduced. In this section we demonstrate how the CE method is applied in order to search for good solutions.

The CE optimization method is an iterative algorithm that consists of two steps:

- Creating a Generation of solutions according to the probability functions and evaluating them.
- Selecting an Elite group that consists of some top quintile of the generation and updating the probability functions according to the values of the variables in the Elite group. The updated probability functions are then used to generate a new Generation.

A chromosome is built of a set of genes. Each gene consists of a sequence of Boolean variables (IN_USE and STOPPING_STATION) and integer variables (EARLIEST_TIME). For each variable a probability function is created. According to this function the variable value will be randomly determined.

Variables distributions

**Boolean variables** - Each variable is assigned with a Bernoulli distribution with a parameter p.

**Integer variables** - For each variable a general discrete probability function is built with a support of \{0,...,CYCLE_TIME-1\}. That is the variable may obtain values ranging from 0 to CYCLE_TIME-1.

Each route has a few possible trains. In order for the information in the genes to be consistent in all chromosomes, we need to make sure that the insertion order of the genes of each route does not change. This is done by separating the supports of EARLIEST_TIME to a number of non-overlapping ranges of approximately equal size. Each possible train is assigned with a different range (see example below).

Initial distribution parameters

Predetermined values of the distribution parameters are given as an input for the creation of the first generation of solutions. All Bernoulli parameters are assigned with the value 0.5.

The distributions of EARLIEST_TIME are initially set to be Uniform, with non-overlapping supports for genes of the same route.

For example, the initial distributions of four possible trains of the same route would be:

- First gene – `EARLIEST_TIME ~ U(0,14)`
- Second gene – `EARLIEST_TIME ~ U(15,29)`
- Third gene – `EARLIEST_TIME ~ U(30,44)`
- Fourth gene - `EARLIEST_TIME ~ U(45,59)`
Updating the distribution parameters

After evaluating all chromosomes in a generation and selecting the elite group, the distribution functions are updated. Recall that in the insertion process some genes having true value of IN_USE may not be eventually scheduled. This raises the question whether or not to regard these genes in the updating process.

Option 1 – updating the distribution functions according to all genes that had true value of IN_USE.
Option 2 – updating the distribution functions only according to genes that were actually scheduled in the insertion process.

In the first numerical experiment that will be presented in the next chapter we test among other things the effect of choosing between one of these two options.

As described in Section 3.6, the updated probability distribution is smoothed in order to prevent the algorithm from a prematurely convergence to a local minimum. The parameter $\alpha$ represents the weight of the current generation and $(1-\alpha)$ represents the weight of the history.

Next we show how the coding of the chromosomes in the elite group is used for updating the parameters. We describe the updating mechanism for each of the components of the genes.

Updating the probability distribution of IN_USE

Option 1 – count the number of times a gene was IN_USE within the solutions of the elite group (num_in_use). The distribution function of IN_USE of gene $i$ at iteration $t$ of the algorithm $IU_{i,t}$ is set according to the following equation:

$$IU_{i,t} = \alpha \cdot \frac{\text{num_in_use}}{\text{elite size}} + (1 - \alpha) \cdot IU_{i,t-1}$$ (22)

Option 2 – count the number of times a gene was scheduled within the solution of the elite group (num_scheduled), update the distribution function according to the following equation:

$$IU_{i,t} = \alpha \cdot \frac{\text{num_scheduled}}{\text{elite size}} + (1 - \alpha) \cdot IU_{i,t-1}$$ (23)

Updating the probability distribution of STOPPING_STATIONS

Taking into consideration genes that have false value of IN_USE may produce wrong information on the proportion of times a train has to stop in a certain station and may damage the convergence of the algorithm. The distribution functions may be updated in the following way:

Option 1:
For a given stop, counting the number of times the variable had true value in the genes that had true value of IN_USE (num_stopping). The distribution function is updated according to the following equation:

$$SS_{i,j,t} = \alpha \cdot \frac{\text{num_stopping}}{\text{num_in_use}} + (1 - \alpha) \cdot SS_{i,j,t-1}$$ (24)

Where $SS_{i,j,t}$ is the distribution parameter of stop $j$ of gene $i$ at iteration $t$ of the algorithm. If num_in_use=0, the parameter is left unchanged.
Option 2:
Counting the number of times the variable had true value in the genes that were actually scheduled in the insertion process (num_stopping). The parameter is updated according to the following equation:

\[ SS_{i,j,t} = \alpha \cdot \frac{num\_stopping}{num\_scheduled} + (1 - \alpha) \cdot SS_{i,j,t-1} \]  

(25)

If num\_scheduled=0, the parameter is left unchanged.

Updating the probability distribution of EARLIEST_TIME

An empirical discrete distribution function based on the relevant values of EARLIEST_TIME is created. In Option 1 we gather the values of EARLIEST_TIME in the genes that had true values of IN_USE. In Option 2 we gather the values of EARLIEST_TIME in genes that where actually scheduled. In both cases the idea is to generate EARLIEST_TIME values that are equal or similar to the gathered values with high probability. To this end we update the distributions such that we obtained general discrete distributions for each route, but we make sure to keep the support of these distributions disjoint while allowing the boundaries between the supports to vary. By doing so, we allow the CE algorithm to explore the entire solution space but prevent switching the orders of train. Such a switch is undesirable because the additional information stored in the genes regarding stopping stations is relevant to a particular train identified by its order. Switching the order of two trains may create completely different solutions that may differ substantially from those found in the elite set.

The above mentioned empirical distribution is based on the elite set and defined as follows. The support of the distribution of the \(i^{th}\) train of a route is between \(a\) and \(b\), where \(a\) is the average between the smallest value of train \(i\) and the largest value of train \(i-1\), and \(b\) is the average between the largest value of train \(i\) and the smallest value of train \(i+1\). The support of the first possible train of a route begins at 0 and respectively the support of the last possible train of a route ends at CYCLE_TIME-1.

For example, consider a route that has three possible trains and assume that in iteration \(t\) of the algorithm the following values of EARLIEST_TIME were gathered from the solutions in the elite group (from solutions with IN_USE=1 in the corresponding trains):

Train 1: \{3, 5, 6, 9, 12\}
Train 2: \{17, 20, 23, 25, 26, 28\}
Train 3: \{43, 45, 53, 54\}

The supports of the discrete distribution functions to be used in the next generation would be:

Train 1: [0,14]
Train 2: [15, 35]
Train 3: [36,59]
Assume that we have N values in the train's list, then the probability of obtaining a value:

- Between \( a \) and the smallest value is \( \frac{1}{2N} \) with equal probability for each integer value in this interval.
- Between the largest value and \( b \) is \( \frac{1}{2N} \) with equal probability for each integer value in this interval.
- The probability of obtaining a value between any pair of values in the list (ordered in non-decreasing order) is \( \frac{1}{N} \) with equal probability for each integer value in this interval.

Train 1 of the above example has 5 values. Hence, the probability of generating a gene with EARLIEST\_TIME between 0 and 3 is \( \frac{1}{10} \), and the probability of generating a gene with EARLIEST\_TIME between 3 and 5 is \( \frac{1}{5} \). Altogether, the probability for the value 2 would be \( \frac{1}{40} \) and the probability for the value 3 would be \( \frac{1}{40} + \frac{1}{15} = \frac{11}{120} \).

Algorithm stopping rules:

First rule: for some small positive value \( \varepsilon \), stop if each Boolean parameter is either less than \( \varepsilon \) or greater than \( 1-\varepsilon \).

Second rule: Stop if the running time of the algorithm has exceeded a certain time limit.

In the next chapter, results and analysis of numerical experiments on a real-world instance of the ILPTP will be presented.

CE algorithm for ILPTP

```plaintext
Input: Initial distribution parameters;
Repeat
    { 
        Generate Generation_size chromosomes according to current Distribution Functions;
        For all chromosomes
            Evaluate chromosome;

        Select Elite group;
        If best chromosome in Elite group has better value than best_chromosome
            Set best_chromosome to best chromosome in Elite group;

        Update Distribution parameters according to Elite Group;
    }
Until convergence is reached or time is exceeded;
Return best_chromosome;
```
6 Numerical Experiments

In this chapter we present a number of experiments that were performed in order to test the algorithm. These experiments aimed to answer the following questions:

- Does the algorithm produce good solutions in reasonable time for ILPTP?
- How sensitive is the output with respect to the algorithm parameters? And how should the parameters be calibrated for solving real-world sized instances?

Numerical results of heuristic methods are typically evaluated by comparison to optimal solutions obtained by exact methods or to lower bounds. It is also common to compare the results to prior results of other solution methods of the same instances. The ILPTP is formulated here for the first time, consequently there are no previous studies to compare with. Because of the complexity of this problem, optimal solutions for real-world instances can not be attained by exact methods, even for small (“toy”) instances. The problem is particularly hard due to the fact that evaluation of any solution requires solving a large scale passenger assignment sub-problem.

In order to benchmark our algorithm we compare our solution of a real-world instance to an existing one, i.e., the current timetable used by a railway operator. Unfortunately, there are no publicly available and acknowledged problem instances for railway scheduling problems, as in several problem areas within the operations research community. Törnquist (2006), identified the need to create such instances and hence we offer a full description of our instance to be used as a benchmark for future line planning and timetabling models and algorithms.

The ILPTP instance that will be presented in this chapter is based on data received from Israel Railways and is equivalent to the problem faced by its planners.

In the rest of this chapter we give some details on the Israeli Railways instance. We will present the design of our experiments and the algorithm’s parameters tested. Later we will present a statistical analysis of the experiments and compare the results to the current timetable used by Israel Railways. In Section 6.4.1, an adaptation of the algorithm to a bi-objective problem that combines both service level and operational costs will be introduced. An efficiency frontier with Pareto-dominating solutions will be presented.
6.1 Benchmark Problem

Israel Railways is an independent government owned corporation. The company provides both freight and passenger transportation and is responsible both for the infrastructure and the rolling stock. Since the entire system is managed by a single organization, a centralistic planning approach can be applied. The infrastructure consists of about 1000 kilometers of rail tracks, 47 passenger stations and 30 operational and freight stations. Some of the blocks are only used by freight trains and therefore are omitted from the following description. The policy of Israel Railways is to give a clear priority to passenger trains even though it might not be as profitable as freight transportation. 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. Freight trains run on blocks that are also used by passenger trains, mainly during the night.

As of 2008, the company operated 14 lines, namely, 14 inbound and 14 outbound routes. Some of these lines are intercity and some are suburban. Note that the published timetable named only 8 lines because some lines that share several segments, but start or ends at different stations, are considered as a single line.

For more information about Israel Railways, see: http://www.rail.co.il/.

The complete data of this instance can be downloaded from: http://www.eng.tau.ac.il/~talraviv/Publications/. An archive file in this address contains several tables and a technical description of their structure. The data files includes: the capacity of each station, the minimal traveling time on each block, the routes used by Israel railways with their priorities and travelling directions and the 2008’s OD matrices.

Figure 21 presents a logical description of the infrastructure used by passenger trains. The figure shows the logical connection between the different blocks and stations, it does not represent the geographical deployment of the infrastructure and it is not drawn to scale.

In our experiments we were interested in checking the ability of the algorithm to produce a better timetable without expending the infrastructure or checking other optional routes. The input consists of the 28 routes used in the 2008’s timetable. Each route was given a priority and the maximal frequency of each route was set to four. Explicitly, a chromosome will consist of 112 genes in different lengths.

The demand data received from Israel Railways aggregates demands for journeys in 2008 partitioned to hours. With guidance from the planners, the data was manipulated to present the demands for journeys in each hour of a regular week-day. The resulting OD matrices contain daily demands for the hours 06:00-01:00 (19 matrices).
Figure 21: Infrastructure representation of the benchmark problem
In Table 9 we present some additional parameters of this instance:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CYCLE_TIME</td>
<td>60 min.</td>
</tr>
<tr>
<td>HORIZON TIME</td>
<td>1140 min.</td>
</tr>
<tr>
<td>EXTRA HEADWAY</td>
<td>1 min.</td>
</tr>
<tr>
<td>BLOCK_UTILIZATION</td>
<td>75%</td>
</tr>
<tr>
<td>STOPPING TIME</td>
<td>2 min.</td>
</tr>
<tr>
<td>TRANSFER_TIME</td>
<td>4 min.</td>
</tr>
<tr>
<td>MAX_DWELLING</td>
<td>30%</td>
</tr>
</tbody>
</table>

The safety and operational parameters EXTRA_HEADWAY and BLOCK_UTILIZATION are set to the same values used by Israel Railways planners. In the timetable planned by Israel Railways, the duration of most of the planned stops is one minute. Taking into consideration the time needed for deceleration and acceleration, at rush hours in many stations this one minute stop may not represent the actual delay caused by stopping at a station. We suggest assuming two minute stopping times. Such a value may allow slacks in some stations that may add to the robustness of the created timetable. Generally, instead of using the same parameter for all stations, one may use a different setting for each station and period of the day. Such a requirement is readily supported by our algorithm. The TRANSFER_TIME parameter is set to four minutes in order to allow safe transferring of all passengers during rush hours and to assure the transfer will still be possible in occurrences of small delays. We note that Israel Railways also uses a four minutes time interval in connection stations, even though transfers are not guaranteed.
6.2 **Experiment Parameters**

In our experiment we tested the following four adjustable algorithm parameters:

Smoothing parameter ($\alpha$) – in the second stage of each CE iteration, the distribution parameters are being updated. As described in Section 5.5, the Bernoulli parameters are updated through an exponentially weighted moving average (smoothing). $\alpha$ represents the weight given to the proportion that was calculated in the current generation. We expect faster convergence for high values of $\alpha$. The values tested in this experiment were 0.3 and 0.7.

Generation size (GS) – The number of solutions to be produced in each iteration of the algorithm. The generation sizes tested in this experiment were 500 and 1000. As recommended in previous studies that applied the CE method, we took elite size to be 10% of the generation size.

Keep Elite (KE) – according to the original definition of CE, see Rubinstein (1997), in each iteration of the algorithm an entirely new set of solutions is produced. In some iterations the Elite group may consist of worse solutions than in previous iterations, this may affect the distribution parameters, the convergence time and the resulting solution. In other evolutionary methods it is common to keep the elite group for the next generation, a practice called elitism. For the cross entropy the whole elite group can be added to the population of the next generation. These solutions compete with all other generated solutions on their place in the current elite set. We tested the effect of applying this idea in our CE algorithm.

Updating distributions according to scheduled trains only (SCH) – in the insertion process some genes having True value of \textsc{IN\_USE} may eventually not be scheduled. As described in Section 5.5, the distribution parameters can be updated according to genes that have True value of \textsc{IN\_USE} (Option 1) or only according to genes that were actually scheduled (Option 2).

6.3 **Statistical Analysis of Parameters Tuning Experiment**

The algorithm that was presented in the previous chapter was implemented in C and was tested on a computer with Intel® Core™ 2 Dou 2.20GHZ CPU and a 64-bit operating system. The algorithm running times presented below should be regarded as "upper bounds", we assume that an experienced programmer could have implemented the algorithm more efficiently.

6.3.1 First Experiment

A full $2^4$ factorial experiment with two replications was carried out. The convergence rule was set to $\varepsilon=0.1$, meaning that the algorithm converges when all Bernoulli parameters are smaller than 0.1 or greater than 0.9. The time limit for each run was set to 12 hours. We believe that this is a modest solution time for such a long run planning problem.
In the following Table we present the experiment results:

**Table 10: First experiment results - objective value, number of generations, running time of each run.**

* indicates runs that reached their 12 hours time limit before convergence

<table>
<thead>
<tr>
<th>α</th>
<th>GS</th>
<th>KE</th>
<th>SCH</th>
<th>Objective (minutes/day)</th>
<th>Generations</th>
<th>Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>500</td>
<td>NO</td>
<td>Option 1</td>
<td>6,530,890</td>
<td>949</td>
<td>* 43,200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6,568,539</td>
<td>1000</td>
<td>* 43,200</td>
</tr>
<tr>
<td>0.7</td>
<td>500</td>
<td>NO</td>
<td>Option 1</td>
<td>6,649,180</td>
<td>842</td>
<td>36,561</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6,707,523</td>
<td>792</td>
<td>35,207</td>
</tr>
<tr>
<td>0.3</td>
<td>1000</td>
<td>NO</td>
<td>Option 1</td>
<td>6,528,516</td>
<td>551</td>
<td>* 43,200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6,538,783</td>
<td>526</td>
<td>* 43,200</td>
</tr>
<tr>
<td>0.7</td>
<td>1000</td>
<td>NO</td>
<td>Option 1</td>
<td>6,582,148</td>
<td>508</td>
<td>* 43,200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6,618,965</td>
<td>457</td>
<td>* 43,200</td>
</tr>
<tr>
<td>0.3</td>
<td>500</td>
<td>Yes</td>
<td>Option 1</td>
<td>6,610,563</td>
<td>1204</td>
<td>* 43,200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6,599,684</td>
<td>1370</td>
<td>* 43,200</td>
</tr>
<tr>
<td>0.7</td>
<td>500</td>
<td>Yes</td>
<td>Option 1</td>
<td>6,603,135</td>
<td>1294</td>
<td>* 43,200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6,623,218</td>
<td>1245</td>
<td>* 43,200</td>
</tr>
<tr>
<td>0.3</td>
<td>1000</td>
<td>Yes</td>
<td>Option 1</td>
<td>6,723,796</td>
<td>789</td>
<td>* 43,200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6,699,712</td>
<td>850</td>
<td>* 43,200</td>
</tr>
<tr>
<td>0.7</td>
<td>1000</td>
<td>Yes</td>
<td>Option 1</td>
<td>6,511,563</td>
<td>643</td>
<td>* 43,200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6,535,434</td>
<td>666</td>
<td>* 43,200</td>
</tr>
<tr>
<td>0.3</td>
<td>500</td>
<td>NO</td>
<td>Option 2</td>
<td>7,025,057</td>
<td>251</td>
<td>5,544</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7,322,705</td>
<td>312</td>
<td>7,604</td>
</tr>
<tr>
<td>0.7</td>
<td>500</td>
<td>NO</td>
<td>Option 2</td>
<td>7,265,468</td>
<td>112</td>
<td>2,557</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7,397,752</td>
<td>113</td>
<td>2,748</td>
</tr>
<tr>
<td>0.3</td>
<td>1000</td>
<td>NO</td>
<td>Option 2</td>
<td>7,267,052</td>
<td>308</td>
<td>15,888</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6,872,812</td>
<td>768</td>
<td>* 43,200</td>
</tr>
<tr>
<td>0.7</td>
<td>1000</td>
<td>NO</td>
<td>Option 2</td>
<td>7,132,093</td>
<td>111</td>
<td>5,000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7,080,117</td>
<td>134</td>
<td>7,005</td>
</tr>
<tr>
<td>0.3</td>
<td>500</td>
<td>Yes</td>
<td>Option 2</td>
<td>6,708,113</td>
<td>1509</td>
<td>* 43,200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6,834,292</td>
<td>481</td>
<td>11,057</td>
</tr>
<tr>
<td>0.7</td>
<td>500</td>
<td>Yes</td>
<td>Option 2</td>
<td>6,921,986</td>
<td>151</td>
<td>2,930</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6,931,104</td>
<td>149</td>
<td>2,908</td>
</tr>
<tr>
<td>0.3</td>
<td>1000</td>
<td>Yes</td>
<td>Option 2</td>
<td>6,715,197</td>
<td>387</td>
<td>18,157</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6,720,951</td>
<td>538</td>
<td>26,944</td>
</tr>
<tr>
<td>0.7</td>
<td>1000</td>
<td>Yes</td>
<td>Option 2</td>
<td>6,898,337</td>
<td>203</td>
<td>7,586</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6,869,580</td>
<td>196</td>
<td>7,872</td>
</tr>
</tbody>
</table>
Statistical Analysis of the experiment:

Table 11: First experiment – Effects estimation

<table>
<thead>
<tr>
<th>Factor</th>
<th>Coefficient Estimate</th>
<th>95% CI Low</th>
<th>95% CI High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>6799821.0</td>
<td>6764205.0</td>
<td>6835437.0</td>
</tr>
<tr>
<td>A</td>
<td>33154.3</td>
<td>-2461.8</td>
<td>68770.3</td>
</tr>
<tr>
<td>GS</td>
<td>-31379.9</td>
<td>-66995.9</td>
<td>4236.025</td>
</tr>
<tr>
<td>KE</td>
<td>-80654.2</td>
<td>-116270.0</td>
<td>-45038.3</td>
</tr>
<tr>
<td>SCH</td>
<td>197842.7</td>
<td>162226.8</td>
<td>233458.7</td>
</tr>
<tr>
<td>α * GS</td>
<td>-23065.8</td>
<td>-58681.7</td>
<td>12550.7</td>
</tr>
<tr>
<td>α * KE</td>
<td>-15526.3</td>
<td>-51142.3</td>
<td>20089.2</td>
</tr>
<tr>
<td>α * SCH</td>
<td>31236.7</td>
<td>-4379.2</td>
<td>66852.6</td>
</tr>
<tr>
<td>GS * KE</td>
<td>21534.4</td>
<td>-14081.5</td>
<td>57150.4</td>
</tr>
<tr>
<td>GS * SCH</td>
<td>-21766.4</td>
<td>-57382.3</td>
<td>13849.7</td>
</tr>
<tr>
<td>KE * SCH</td>
<td>-92064.2</td>
<td>-127680.0</td>
<td>-56448.3</td>
</tr>
<tr>
<td>α * GS * KE</td>
<td>-155.0</td>
<td>-35771.0</td>
<td>35460.9</td>
</tr>
<tr>
<td>α * GS * SCH</td>
<td>9188.9</td>
<td>-26427.0</td>
<td>44804.2</td>
</tr>
<tr>
<td>α * KE * SCH</td>
<td>31441.9</td>
<td>-4173.95</td>
<td>67057.3</td>
</tr>
<tr>
<td>GS * KE * SCH</td>
<td>7682.8</td>
<td>-27933.1</td>
<td>43298.1</td>
</tr>
<tr>
<td>α * GS * KE * SCH</td>
<td>16667.3</td>
<td>-18948.6</td>
<td>52283.3</td>
</tr>
</tbody>
</table>

Table 12: First experiment - ANOVA

<table>
<thead>
<tr>
<th>Source</th>
<th>Sum of Squares</th>
<th>df</th>
<th>Mean Square</th>
<th>F Value</th>
<th>p-value Prob &gt; F</th>
<th>Effects Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>1.93E+12</td>
<td>15</td>
<td>1.29E+11</td>
<td>14.24</td>
<td>&lt; 0.0001</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>3.52E+10</td>
<td>1</td>
<td>3.52E+10</td>
<td>3.89</td>
<td>0.0660</td>
<td>1.70%</td>
</tr>
<tr>
<td>GS</td>
<td>3.15E+10</td>
<td>1</td>
<td>3.15E+10</td>
<td>3.49</td>
<td>0.0802</td>
<td>1.52%</td>
</tr>
<tr>
<td>KE</td>
<td>2.08E+11</td>
<td>1</td>
<td>2.08E+11</td>
<td>23.05</td>
<td>0.0002</td>
<td>10.04%</td>
</tr>
<tr>
<td>SCH</td>
<td>1.25E+12</td>
<td>1</td>
<td>1.25E+12</td>
<td>138.67</td>
<td>&lt; 0.0001</td>
<td>60.39%</td>
</tr>
<tr>
<td>α * GS</td>
<td>1.7E+10</td>
<td>1</td>
<td>1.7E+10</td>
<td>1.88</td>
<td>0.1887</td>
<td>0.82%</td>
</tr>
<tr>
<td>α * KE</td>
<td>7.71E+09</td>
<td>1</td>
<td>7.71E+09</td>
<td>0.85</td>
<td>0.3691</td>
<td>0.37%</td>
</tr>
<tr>
<td>α * SCH</td>
<td>3.12E+10</td>
<td>1</td>
<td>3.12E+10</td>
<td>3.46</td>
<td>0.0815</td>
<td>1.51%</td>
</tr>
<tr>
<td>GS * KE</td>
<td>1.48E+10</td>
<td>1</td>
<td>1.48E+10</td>
<td>1.64</td>
<td>0.2182</td>
<td>0.72%</td>
</tr>
<tr>
<td>GS * SCH</td>
<td>1.52E+10</td>
<td>1</td>
<td>1.52E+10</td>
<td>1.68</td>
<td>0.2135</td>
<td>0.73%</td>
</tr>
<tr>
<td>KE * SCH</td>
<td>2.71E+11</td>
<td>1</td>
<td>2.71E+11</td>
<td>30.03</td>
<td>&lt; 0.0001</td>
<td>13.08%</td>
</tr>
<tr>
<td>α * GS * KE</td>
<td>769171.3</td>
<td>1</td>
<td>7.69E+05</td>
<td>0.00</td>
<td>0.9928</td>
<td>0.00%</td>
</tr>
<tr>
<td>α * GS * SCH</td>
<td>2.7E+09</td>
<td>1</td>
<td>2.7E+09</td>
<td>0.30</td>
<td>0.5920</td>
<td>0.13%</td>
</tr>
<tr>
<td>α * KE * SCH</td>
<td>3.16E+10</td>
<td>1</td>
<td>3.16E+10</td>
<td>3.50</td>
<td>0.0797</td>
<td>1.53%</td>
</tr>
<tr>
<td>GS * KE * SCH</td>
<td>1.89E+09</td>
<td>1</td>
<td>1.89E+09</td>
<td>0.21</td>
<td>0.6536</td>
<td>0.09%</td>
</tr>
<tr>
<td>α * GS * KE * SCH</td>
<td>8.89E+09</td>
<td>1</td>
<td>8.89E+09</td>
<td>0.98</td>
<td>0.3359</td>
<td>0.43%</td>
</tr>
<tr>
<td>Pure Error</td>
<td>1.45E+11</td>
<td>16</td>
<td>9.03E+09</td>
<td>9.03</td>
<td>6.97%</td>
<td></td>
</tr>
<tr>
<td>Cor Total</td>
<td>2.07E+12</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
From the above ANOVA analysis of the factorial experiment we can see that the SCH factor had a great effect on the results. More specifically, all runs where the distributions parameters were updated only according to scheduled genes (SCH=1) converged to local optima in a few hours (3.5 hours in average). On the other hand, almost all runs with SCH=0 did not converge in 12 hours and more importantly these runs produced significantly better results.

6.3.2 Second Experiment

Since the effect of SCH parameter seems dramatic as compared to the rest of the effects, we conclude that Option 1 should be used, namely, the Bernoulli distribution parameters should be updated based on all genes that have a true value of IN_USE. It seems that ignoring trains that could not be scheduled may prematurely disqualify too many trains out of the generated solutions.

Our first experiment is not conclusive about the other parameters. We point out that although the p-value of KE is very small, the KE effect is much smaller than the SCH effect and there is a large negative interaction with SCH that cancels the main KE effect. Next, we check the rest of the three effects based on a full 2^3 factorial experiment with four replications. For each run we used two replications from the first experiment and two additional replications were executed. The combined results are presented Table 13.

The ANOVA analysis (Table 15) shows that the main factors did not have significant effects. In addition, the gap between the best and the worse solution obtained is less than 4%. This may suggest that the algorithm is not sensitive to changes in the settings of these parameters. However, our experiment implies that the following algorithm parameters may work best

- \(\alpha=0.7, \text{GS}=1000, \text{KE}=\text{YES}\)
- \(\alpha=0.3, \text{GS}=1000, \text{KE}=\text{NO}\)

We note that it might be the case that this finding is specific to the Israeli Railways instance. In particular, we suspect that the generation size parameter (GS) should be selected proportionally to the problem dimension.

The effects of all the interactions were relatively small but significant. However, the largest effect in this model was the interaction between \(\alpha\) and KE. These two factors represent, in some sense, the weights of the previous generation, as compared to the current ones. The smoothing parameter \(\alpha\) does this in a direct way, i.e., a smaller value of \(\alpha\) puts more weight on the previous generation; Elitism allows the best solutions of the previous generation to survive and to affect the current generation. Hence, better results were obtained while giving high weight to new generations (\(\alpha=0.7\)) and keeping the elite group, or while giving low weight to new generations (\(\alpha=0.3\)) and not keeping the elite group. This may suggest that these two parameters balance each other so that new generations are not given excessively high or low weights. It is not clear whether keeping the elite group has another contribution to the algorithm or it only acts as a different way to give a larger weight to previous generations. The effect of the interaction between \(\alpha\) and GS is also small but significant. As the generation size grows, more weight should be given to the current generation. Combination of high \(\alpha\) and small generation size may lead to premature convergence to a local minimum.
Table 13: Second experiment results - objective value, number of generations, running time of each run. * indicates runs that reached their 12 hours time limit before convergence

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>GS</th>
<th>KE</th>
<th><strong>Objective (minutes/day)</strong></th>
<th><strong>Generations</strong></th>
<th><strong>Time (seconds)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>500</td>
<td>No</td>
<td>6,539,744</td>
<td>1019</td>
<td>*43,200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6,617,744</td>
<td>1039</td>
<td>*43,200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6,530,890</td>
<td>949</td>
<td>*43,200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6,568,539</td>
<td>1000</td>
<td>*43,200</td>
</tr>
<tr>
<td>0.7</td>
<td>500</td>
<td>No</td>
<td>6,651,948</td>
<td>1040</td>
<td>*43,200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6,702,584</td>
<td>683</td>
<td>30,509</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6,649,180</td>
<td>842</td>
<td>36,561</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6,707,523</td>
<td>792</td>
<td>35,207</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>No</td>
<td>6,577,572</td>
<td>538</td>
<td>*43,200</td>
</tr>
<tr>
<td>0.3</td>
<td></td>
<td></td>
<td>6,480,812</td>
<td>548</td>
<td>*43,200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6,528,516</td>
<td>551</td>
<td>*43,200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6,538,783</td>
<td>526</td>
<td>*43,200</td>
</tr>
<tr>
<td>0.7</td>
<td>1000</td>
<td>No</td>
<td>6,583,713</td>
<td>538</td>
<td>*43,200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6,565,425</td>
<td>544</td>
<td>*43,200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6,582,148</td>
<td>508</td>
<td>*43,200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6,618,965</td>
<td>457</td>
<td>*43,200</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>Yes</td>
<td>6,597,762</td>
<td>1269</td>
<td>*43,200</td>
</tr>
<tr>
<td>0.3</td>
<td></td>
<td></td>
<td>6,535,035</td>
<td>1232</td>
<td>*43,200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6,610,563</td>
<td>1204</td>
<td>*43,200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6,599,684</td>
<td>1370</td>
<td>*43,200</td>
</tr>
<tr>
<td>0.7</td>
<td>500</td>
<td>Yes</td>
<td>6,580,638</td>
<td>1136</td>
<td>*43,200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6,549,768</td>
<td>1124</td>
<td>*43,200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6,603,135</td>
<td>1294</td>
<td>*43,200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6,623,218</td>
<td>1245</td>
<td>*43,200</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>Yes</td>
<td>6,691,339</td>
<td>816</td>
<td>*43,200</td>
</tr>
<tr>
<td>0.3</td>
<td></td>
<td></td>
<td>6,744,589</td>
<td>827</td>
<td>*43,200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6,723,796</td>
<td>789</td>
<td>*43,200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6,699,712</td>
<td>850</td>
<td>*43,200</td>
</tr>
<tr>
<td>0.7</td>
<td>500</td>
<td>Yes</td>
<td>6,496,337</td>
<td>748</td>
<td>*43,200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6,524,078</td>
<td>681</td>
<td>*43,200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6,511,563</td>
<td>643</td>
<td>*43,200</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>Yes</td>
<td>6,535,434</td>
<td>666</td>
<td>*43,200</td>
</tr>
</tbody>
</table>
Statistical Analysis of the experiment:

**Table 14: Second experiment – Effects estimation**

<table>
<thead>
<tr>
<th>Factor</th>
<th>Coefficient Estimate</th>
<th>95% CI Low</th>
<th>95% CI High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>6595960.0</td>
<td>6584678.0</td>
<td>6607243.0</td>
</tr>
<tr>
<td>α</td>
<td>-3107.0</td>
<td>-14389.6</td>
<td>8175.6</td>
</tr>
<tr>
<td>GS</td>
<td>-8286.7</td>
<td>-19569.4</td>
<td>2995.8</td>
</tr>
<tr>
<td>KE</td>
<td>5705.1</td>
<td>-5577.4</td>
<td>16987.7</td>
</tr>
<tr>
<td>α * GS</td>
<td>-32359.0</td>
<td>-43641.6</td>
<td>-21076.4</td>
</tr>
<tr>
<td>α * KE</td>
<td>-45537.4</td>
<td>-56820.0</td>
<td>-34254.8</td>
</tr>
<tr>
<td>GS * KE</td>
<td>22476.9</td>
<td>11194.3</td>
<td>33759.6</td>
</tr>
<tr>
<td>α * GS * KE</td>
<td>-17999.7</td>
<td>-29282.3</td>
<td>-6717.0</td>
</tr>
</tbody>
</table>

**Table 15: Second experiment – ANOVA**

<table>
<thead>
<tr>
<th>Source</th>
<th>Sum of Squares</th>
<th>df</th>
<th>Mean Square</th>
<th>F Value</th>
<th>p-value Prob &gt; F</th>
<th>Effects Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>1.3E+11</td>
<td>7</td>
<td>1.86E+10</td>
<td>19.41</td>
<td>&lt; 0.0001</td>
<td></td>
</tr>
<tr>
<td>α</td>
<td>3.09E+08</td>
<td>1</td>
<td>3.09E+08</td>
<td>0.32</td>
<td>0.5751</td>
<td>0.20%</td>
</tr>
<tr>
<td>GS</td>
<td>2.2E+09</td>
<td>1</td>
<td>2.2E+09</td>
<td>2.30</td>
<td>0.1426</td>
<td>1.44%</td>
</tr>
<tr>
<td>KE</td>
<td>1.04E+09</td>
<td>1</td>
<td>1.04E+09</td>
<td>1.09</td>
<td>0.3071</td>
<td>0.68%</td>
</tr>
<tr>
<td>α * GS</td>
<td>3.35E+10</td>
<td>1</td>
<td>3.35E+10</td>
<td>35.04</td>
<td>&lt; 0.0001</td>
<td>21.91%</td>
</tr>
<tr>
<td>α * KE</td>
<td>6.64E+10</td>
<td>1</td>
<td>6.64E+10</td>
<td>69.39</td>
<td>&lt; 0.0001</td>
<td>43.40%</td>
</tr>
<tr>
<td>GS * KE</td>
<td>1.62E+10</td>
<td>1</td>
<td>1.62E+10</td>
<td>16.91</td>
<td>0.0004</td>
<td>10.57%</td>
</tr>
<tr>
<td>α * GS * KE</td>
<td>1.04E+10</td>
<td>1</td>
<td>1.04E+10</td>
<td>10.84</td>
<td>0.0031</td>
<td>6.78%</td>
</tr>
<tr>
<td>Pure Error</td>
<td>2.3E+10</td>
<td>24</td>
<td>9.56E+08</td>
<td>15.01%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cor Total</td>
<td>1.53E+11</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6.3.3 Convergence of the Algorithm

Next we look into the convergence process of our algorithm and try to derive some insights. Figure 22 represents the development of the best solution obtained in this experiment. As can be seen in this Figure the significant improvement was achieved in approximately two hours. In about 6 hours the algorithm almost reached the best obtained solution but the convergence rule used by our algorithm was not triggered. In fact the algorithm was stopped due to time limit rather than triggering the convergence rule in most cases, see Table 13. This may be explained by the following property of the solution encoding. A gene with a very low priority may have the same effect whether it has True or False values of IN_USE (since there is a good chance it will not be scheduled in the insertion process). Therefore, in some good solutions this gene may have True values of IN_USE and in others False. Consequently, the relevant distribution parameter may never converge.

![Figure 22: Convergence of the best solution in the second experiment](image)

In Table 16 we present for each run the first time that the best solution in that run was produced (the time column represents the running time of each run). In addition, we present the first time when solutions with a 0.5% and 1% gap from the best solution in the run were produced. On average the algorithm reached 1% of the best solution in each run in less than 4.5 hours.
Table 16: Solutions progress - running time of each run, first time to best solution, first time to 0.5% from best solution obtained in run, first time to 0.5% from best solution obtained in run.

<table>
<thead>
<tr>
<th>α</th>
<th>GS</th>
<th>KE</th>
<th>Time (seconds)</th>
<th>First time to best</th>
<th>0.5%</th>
<th>1%</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>500</td>
<td>No</td>
<td>*43,200</td>
<td>42,961</td>
<td>9,950</td>
<td>8,098</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*43,200</td>
<td>27,532</td>
<td>15,822</td>
<td>8,848</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*43,200</td>
<td>34,895</td>
<td>9,521</td>
<td>8,939</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*43,200</td>
<td>20,035</td>
<td>12,948</td>
<td>9,656</td>
</tr>
<tr>
<td>0.7</td>
<td>500</td>
<td>No</td>
<td>*43,200</td>
<td>16,237</td>
<td>5,338</td>
<td>3,211</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>30,509</td>
<td>12,502</td>
<td>4,184</td>
<td>3,517</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>36,561</td>
<td>35,252</td>
<td>4,797</td>
<td>3,908</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>35,207</td>
<td>33,279</td>
<td>8,254</td>
<td>3,771</td>
</tr>
<tr>
<td>0.3</td>
<td>1000</td>
<td>No</td>
<td>*43,200</td>
<td>40,347</td>
<td>18,811</td>
<td>16,565</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*43,200</td>
<td>37,359</td>
<td>21,256</td>
<td>19,659</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*43,200</td>
<td>37,418</td>
<td>19,577</td>
<td>16,642</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*43,200</td>
<td>43,101</td>
<td>16,831</td>
<td>15,939</td>
</tr>
<tr>
<td>0.7</td>
<td>1000</td>
<td>No</td>
<td>*43,200</td>
<td>30,267</td>
<td>9,280</td>
<td>7,347</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*43,200</td>
<td>31,203</td>
<td>8,639</td>
<td>7,444</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*43,200</td>
<td>29,844</td>
<td>9,785</td>
<td>8,067</td>
</tr>
<tr>
<td>0.3</td>
<td>500</td>
<td>Yes</td>
<td>*43,200</td>
<td>37,336</td>
<td>19,929</td>
<td>16,980</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*43,200</td>
<td>42,482</td>
<td>23,216</td>
<td>16,661</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*43,200</td>
<td>41,516</td>
<td>18,707</td>
<td>17,649</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*43,200</td>
<td>39,260</td>
<td>24,232</td>
<td>22,458</td>
</tr>
<tr>
<td>0.7</td>
<td>500</td>
<td>Yes</td>
<td>*43,200</td>
<td>27,097</td>
<td>7,980</td>
<td>7,177</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*43,200</td>
<td>42,547</td>
<td>13,592</td>
<td>12,978</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*43,200</td>
<td>21,779</td>
<td>14,890</td>
<td>12,594</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*43,200</td>
<td>23,536</td>
<td>10,678</td>
<td>10,029</td>
</tr>
<tr>
<td>0.3</td>
<td>1000</td>
<td>Yes</td>
<td>*43,200</td>
<td>38,317</td>
<td>32,536</td>
<td>27,700</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*43,200</td>
<td>42,881</td>
<td>34,170</td>
<td>23,975</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*43,200</td>
<td>42,294</td>
<td>39,256</td>
<td>39,131</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*43,200</td>
<td>41,034</td>
<td>37,377</td>
<td>29,207</td>
</tr>
<tr>
<td>0.7</td>
<td>1000</td>
<td>Yes</td>
<td>*43,200</td>
<td>42,386</td>
<td>35,671</td>
<td>33,476</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*43,200</td>
<td>42,984</td>
<td>25,355</td>
<td>23,858</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*43,200</td>
<td>43,085</td>
<td>29,410</td>
<td>28,436</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*43,200</td>
<td>43,189</td>
<td>25,127</td>
<td>24,336</td>
</tr>
</tbody>
</table>

We note that a chromosome of this instance consists of 1480 Boolean Variables (112 IN_USE, 1368 STOPPING_STATIONS). It might be the case that almost all the parameters of the Bernoulli probability distributions of these variables were converged to zero or one, but the convergence rule was not triggered because of very few such distributions that have not converged. Indeed, in some cases the algorithm was stopped after 12 hours, although it reached the best solution a long time before. This may suggest that a different or additional convergence rule should be used, for example: stop if there was no improvement in the value of the best solution for some predefined number of generations.
In spite of the above, recall that a planned timetable would be usually used for a long period, varying from several months to a few years. Since the planning problem needs to be re-solved only after a long time period, the running time of the algorithm becomes less significant. In other words, it makes no difference whether the algorithm runs for a few hours or for a few days. In the best settings (i.e., (α=0.7, GS=1000, KE=YES) and (α=0.3, GS=1000, KE=NO)), the best timetable of each run was produced a short time before the 12 hour limit. This may suggest that letting the algorithm run for longer time periods would have led to somewhat better timetables. For other instances we propose taking the following approach: Start with relatively short time-limits to screen-out inferior settings, those that quickly converged to local minima. Next, let the algorithm with the more promising parameters setting to continue running for longer time periods.

6.3.4 Comparing CE with a Brute Force Approach

One criticism on the CE algorithm may be that the improvement of the solutions results from the fact that we generate more and more solutions. In other words, one may argue that the updating mechanism of the distributions parameters does not contribute to the development of better solutions. In order to refute this argument, the following test was performed: we used the initial CE distributions to randomly generate and evaluate as many feasible solutions as possible within a time limit of 12 hours. This is in fact equivalent to running our algorithm with α=0. We replicated this test four times. The best results of these runs are presented in Table 17. There is a substantial difference between these results and the results obtained by our algorithm, with approximately 20% difference between the averages. We note that the average here is compared to the average over all our runs with all the parameter combinations, as presented in Table 13. If we compare it to the average of the best setting (i.e. α=0.7, GS=1000, KE=True) then the difference is 21.5%.

Table 17: Results of brute force runs

<table>
<thead>
<tr>
<th>α</th>
<th>GS</th>
<th>KE</th>
<th>Objective (minutes)</th>
<th>Generations</th>
<th>Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1000</td>
<td>0</td>
<td>7,938,458</td>
<td>1035</td>
<td>43,222</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7,958,071</td>
<td>1127</td>
<td>43,230</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7,870,806</td>
<td>1124</td>
<td>43,220</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7,902,186</td>
<td>1107</td>
<td>43,224</td>
</tr>
</tbody>
</table>
6.4 Comparison to Current Timetable

ILPTP is formulated for the first time in this work, therefore we can not compare our results to results obtained in previous studies. We offer this instance and the received results as a benchmark for future studies of this problem. Generating an optimal solution for this instance using exact methods seems impractical. Moreover, the need to assign each flow of passengers with an optimal journey makes even "toy" problems hard to solve. The self evident comparison would be to an existing timetable that is used in practice. We compare our result to the timetable published by Israel Railways in March 2008 and was used until January 2010.

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.

The published timetable and the "Full" timetable were evaluated according to the objective function (using the reachability algorithm). The results are presented in the following table:

<table>
<thead>
<tr>
<th>Table 18: Comparison to current timetable – timetables values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Total Passenger Travel Time</strong></td>
</tr>
<tr>
<td>Israel Railways 2008</td>
</tr>
<tr>
<td>Israel Railways &quot;Full&quot;</td>
</tr>
<tr>
<td>CE best Solution</td>
</tr>
</tbody>
</table>

In the solution obtained by the CE algorithm, the average time a passenger spends in the railway system, is reduced by 20%. A simple (but very loose) lower bound can be calculated in the following way: Assuming each passenger is served at the moment of entry to the origin station and that a direct train is used to follow the shortest possible route. The total travel time of each flow of passengers can be calculated by summing the minimal travel times on the blocks that connect the origin and destination. The value of this bound for the Israeli Railway instance is 67,497.8 Hours. This amount of time will be spent in the system in any planned timetable. Referring only to the remainder, the amount of time reduced in the solution of the CE algorithm is about 40%.

The Ayalon corridor is a 7 kilometer section connecting the Tel-Aviv stations, all segments in this section consist of three blocks. This corridor is considered by the railway planners as the system's "bottle-neck", see for example Kedmi (2006, February 21). The CE algorithm was able to schedule substantially more trains in this corridor.

<table>
<thead>
<tr>
<th>Table 19: Comparison to current timetable – trains per hour on Ayalon corridor</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Trains Per Hour on Ayalon Corridor</strong></td>
</tr>
<tr>
<td>Israel Railways &quot;Full&quot;</td>
</tr>
<tr>
<td>CE best Solution</td>
</tr>
</tbody>
</table>
Nowadays, Israel Railways is promoting a 2 billion NIS project for the development of a fourth rail in this corridor (http://www.rail.co.il/HE/Development/Future/). As can be seen, a higher utilization of this corridor can be achieved without adding another rail. Our algorithm can be used to test the effect of adding another rail in the corridor, or alternatively the effect of only adding platforms in some stations in the corridor.

**Bi-objective Problem**
The main objective of this study is to create timetables with high service levels. So far, operational costs were neglected. Naturally, the algorithm has tried to schedule as many trains as possible. Some of the improvement in the service level results from the fact that the algorithm has succeeded in scheduling more trains compared to the current timetable.

We adopted our algorithm to solve a bi-objective problem to explore the tradeoff between operational costs and service level. Operational costs are represented by the total travel time of the trains. In order to construct a timetable that balances the operational costs and the service level, the ratio between these costs needs to be known. The question is how many passenger hours are worth an additional train operation hour? Once this question is answered, the adaptation of our CE algorithm is straightforward. The total number of train hours of each generated schedule can be easily and quickly calculated and its weighted value can be added to the evaluation of the solutions. The CE method will then produce solutions that balance service level vs. operational cost.

We used this idea to construct an efficiency frontier on the service level – operational cost axes. An “optimal” weighted solution was calculated for several weight values. We used our CE algorithm with the following parameter setting: \( \alpha=0.7, \) GS=1000, KE=YES. Each such solution was then broken down back to its service level and operational cost components. The blue line in Figure 23 describes the efficiency frontier obtained with our original problem parameters. Note that we set the STOPPING TIME parameter to be two minutes in order to create more robust timetables. However, for the sake of comparison, since the Israel Railways planners assume one minute stopping time, we also constructed an efficiency frontier based on this value. This frontier is presented as a green curve on the graph. As expected, relaxing the STOPPING_TIME restriction resulted in even better schedules. The current (published) timetable is shown on the graph as a blue rhombus and the “full” timetable as a red square. It is apparent that even the more conservative problem setting allows many schedules that dominate the current solution, both in terms of service level and operational costs.
Figure 23: Efficiency frontiers
7 Conclusions

The railway planning problem consists of several interrelated sub-problems. In this work we dealt with two of the main sub-problems: The Line Planning problem and The Train Timetabling problem. Each of these problems has received great attention in the literature. Since both problems are hard to solve, they are typically solved separately in a hierarchic order, so that the solution for The Line Planning Problem is a part of The Train Timetabling problem input. Very few studies deal with these two problems simultaneously.

We note that the algorithm presented in this study may not replace experienced planners, but it can definitely assist them with identifying better initial solutions for implementation.

7.1 Contribution of this study

This work focuses on improving the service level given to the passengers. The service level is measured by the total time that the passengers spend in the railway system. The integration of the two main problems enables examining more coordination possibilities between the different lines, which consequently reduces the time spent by passengers in the system. We note that this is the first work to represent service this way. Previous studies considered the service level more simplistically at the line planning phase.

The study presents a new model for Integrated Line Planning and Timetabling (ILPT) and formally defines a corresponding optimization problem (ILPTP). Our model can capture general and realistic railway systems. In particular, all types of infrastructure discussed in the literature fit into our model.

A Cross Entropy heuristic is developed for the ILPTP and applied successfully to a large scale real life instance that represents the Israeli railway system. This is the first application of CE in the railway planning domain. We found the CE meta-heuristic particularly suitable for this complex problem.

A compact encoding method for solutions of the ILPTP was presented. Later, an algorithm for decoding a solution to a cyclic timetable was developed. The decoding phase consists of an insertion process similar to the one used in manual planning. This process creates a single feasible cycle that is later repeated over the planning horizon, so that a cyclic timetable is constructed. The insertion process is designed to create solutions that satisfy all the safety and operational restrictions. The line plan is also decided upon during this process. Namely, the serviced lines are selected from a predefined menu and the frequency of each line is determined. In addition, the stopping stations of each occurrence of each line are determined. We note that since our model uses an objective function that is simultaneously affected by decisions related to both the line planning and timetabling, it is the first to truly integrate both components of the problem.

We show analytically that the ILPTP model admits the “maximal speed” property. Namely, under some reasonable assumptions, there is always an optimal solution where all trains are scheduled to traverse each block in minimal time.
Each timetable generated by the algorithm is evaluated based on the time spent by the passengers in the system. This requires finding the shortest journey for each flow of passengers. We construct a graph representation of a timetable such that the shortest journey (including waiting in origin and transfer stations along the way) is equivalent to a shortest path on the graph. We then devise a specialized efficient shortest path algorithm for this graph that is used to quickly evaluate the objective function. Our algorithm runs in linear time with respect to the number of arrival/departure events and to the number of passenger stations. This algorithm is significantly faster than the standard algorithms for all-pairs shortest paths. We note that this algorithm may be used also within a Decision Support System in order to evaluate manually created timetables.

Although the evaluation of the timetable is done with a very efficient algorithm, this part consumes the major portion of the computation time. Choosing a different objective function may lead to a significantly faster algorithm. However, we believe that total travel time of passengers is the proper way to represent the quality of service.

The ILPTP model includes realistic safety and operations constraints regarding seizing of blocks, headway restrictions, capacity of stations and other constraints. Unlike many previous studies, this study presents an algorithm that is readily applicable to real world problems.

An instance of ILPTP based on data received from Israel Railways was presented. A factorial experiment was built to tune the CE algorithm parameters. The results show that the performances of the algorithm are not sensitive to the value of the parameters.

We showed that solutions obtained from our CE algorithm are significantly better than ones obtained by using a brute force randomized approach with equivalent computational effort. In addition, a comparison to the 2008 timetable of Israel Railways was conducted. The best result obtained by the algorithm was 20% better than the value of the actual 2008 timetable. This represents an average saving of more than 6,750,000 travelling hours a year, and shortening the average journey of a passenger from 62 to 50 minutes. The actual utility of such an improvement may extend, indirectly, to many other commuters. For examples, some commuters may start using the train rather than other modes of transportation and some others will continue to use their cars but will enjoy less congested roads. The positive environmental consequences are also clear.

A bi-objective version of the model was formulated taking into account both service level and operational costs. Our algorithm can produce solutions that strongly dominate the timetable used by Israel Railways in 2008 in terms of both the service level and the operational cost.
7.2 Recommendations for future research

We have shown that it is always worthwhile to modify a given schedule so that all trains are scheduled to travel at their maximal speed. A game plan for the insertion process that guarantees this was introduced. Other fine-tunings may be considered.

Our algorithm employs an insertion process to generate feasible solutions based on their randomly generated representations. We note that further polishing of the generated solution may result in convergence to better final results. For example, it may be beneficial to try to move dwelling times at sidings to preceding stations in the route. Such a modification can only reduce the total travel time of the passengers.

To demonstrate this we look at a feasible solution previously presented in Section 5.2.2 (Figure 17), in Figure 24 below. Note that Train 4 dwells in the siding after block 2002 for four minutes starting at time 37. It may be possible to let the train dwell for an additional two minutes (yellow rectangle) in the previous station (prior to block 2002), instead of dwelling in this siding. Such a modification may allow more passengers to "catch" the train at this station while the total travel time of all passengers on board is not affected. The polishing procedure may be applied either to each generated solutions, to those in or close to the elite set, or even only to the final solution delivered from the algorithm.

The demand for journeys is typically non-homogenous throughout the planning horizon and is usually characterized by rush-hours and slack-hours. Hence, one may consider a planning process in which some time periods are planned separately and later the joint timetable is evaluated.

Alternatively, a planning process similar to the one used by Israel Railways may be considered. A full cyclic timetable is planned and afterwards some trains in the slack-hours are removed. The encoding method introduced in this study can be adjusted by adding another Boolean variable to each gene, representing whether the train is used in slack hours or not. The time intervals of the slack hours can be defined separately for each line.
In the algorithm implementation presented in this work, the train's priorities for insertion were given as an input. Alternatively, the priorities could be determined by the CE mechanism.

We note that implementation of all the ideas discussed above is straightforward (though may be tedious). An extensive numerical study is needed to determine which of these modifications may lead to more effective algorithms and thus worthwhile to be implemented.

Finally, in this study two major components of the railway planning process were integrated. Further integration that may include, for example, Platforming and Rolling Stock Circulation, should be considered.
References:


בעיית התכנון במערכת רכבות נוסעים ניתנת לחלוקה למספר תתי בעיות עיקריות: תכנון קווי הנסיעה, תכנון לוח הזמנים, שיבוץ רכבות לרציפים בתחנות, שיבוץ מערכות רכבת ושיבוץ צוותי עובדים. המ不得转载 את כל אחת מהבעיות נמצאת בפני עצמה, כך שהפתרון של בעיה אחת משמש כקלט לבעיה הבאה. מכיוון שהפתרון אינו אינטגרטיבי קיימת סבירות גבוהה כי פתרונות טובים לבעיית התכנון הכוללת יאבדו." בעיות המתחברות בין קווי הנסיעה ואיתוך לוח הזמנים הופכות לבעיה שהינה אף קשה יותר לפתרון. המחקר מתמקד בטייב השירות שניתן לנוסעים במערכת. מודל משולב של שתי הבעיות מאפשר לנו להציג מדד חדש לטיב השירות שאינו קיים בספרות המחקר הרלוונטית. כמקובל במערכות שירות רבות, אנו ממדים את טיב השירות על פיCKET CLOCK, יאבדו בדרך. לברר, מודל משולב של שתי הבעיות יוצר בעיה שהינה אף קשה יותר לפתרון. בפערת המחקר הציגנו מודל של בעיה המשלבת את תכנון קווי הנסיעה ואת תכנון לוח הזמנים. בניית תשתית המערכת (מסילות ותחנות) ואת ביקושי הנסיעה ב-2008 בחלוקה לזוגות מוצא-יעד ובחלוקה לשעות היום. לוח הזמנים הטוב ביותר שהתקבל במסגרת הרצות של האלגוריתם מעלה יותר מ-20% את סך זמן השתייה של הנוסעים במערכת. קקטון זה מייצג חסוך של למעלה מ-6,750,000 שעות אדם בשנה. האלגוריתם שפותח מייצר לוח זמנים ותוכנית קווים בתווכת מחשב בהשקעה של חודשי אדם רבים. בירוק, האלגוריתם הותקן ושוכן בבתי הדואר ובתי מגורים במגוון מחלקות. בנחיתת ו yoğunות הנסיעה של הלוח הזמנים קיים הגה долго זמן וררים גם ב الفنان מלהקה הופעות. י démarch替え המודל של התיאום בין הלוח הזמנים של הרכבות והשירותים המשולבים עם הרכבות.
הכותב והמחמוד מוקדני שירוח של קוני רבבות בגוונים

חברת זה הוגש בעבודת גמר lakes ציchlor החנואר "מקסיקט אוניברסיטאות" בהנדיםית תעשיות
על - ד"ר

מור כספי
העבירה נועשה בראשות מחכלכ החנושת תעשיות
בניהות ד"ר טלי רבין

אסייר תשעים
העברות והזמנות מבוקשים לניירות
כותב רכבות נוסעים

חיבור זה הוגש כעבודה גמר לكرיאת התואר "מוסמך אוניברסיטה" בניהול תקשורת עיתונאית
על – 디
Mor Caspi

ediatorת נועש