Simulation Modelling Practice and Theory 19 (2011) 393–405

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

An event-driven simulator for multi-line metro systems and its application to Santiago de Chile metropolitan rail network

Pablo Grube, Felipe Núñez, Aldo Cipriano *

College of Engineering, Pontificia Universidad Católica de Chile, Santiago, Chile

ARTICLE INFO

Article history:
Received 29 December 2009
Received in revised form 25 July 2010
Accepted 28 July 2010
Available online 7 August 2010

Keywords:
Public transportation
Metro systems
Computer aided simulation
Event-driven simulation
Feedback control

ABSTRACT

Metros are the principal means of public transportation in many of the world's cities, and continue to grow in the face of rising demand. Expanding metro infrastructure is costly, however, and at a certain point becomes unsustainable. When this occurs the only feasible solution is to improve the train's management system by using either offline approaches, such as pre-programming schedules which use historic information, or online approaches which employ system status information obtained during operation. A new planning or control system, be it on or off line, requires prior testing that usually involves conducting simulations. This paper presents the design and implementation of an event-driven dynamic simulator for multi-line metro systems, and its practical application for studying different operating strategies. The simulator is based on object-oriented programming and is capable of interacting with Matlab programs written by the user to design and evaluate real-time control strategies. This article describes the model upon which the simulator is based, presents the user interface, and demonstrates how to use the simulator for operating strategies evaluation in the Santiago de Chile multi-line metropolitan rail network.

1. Introduction

1.1. Motivation

Transportation has become a major issue all around the world. Increased car usage has faced cities with the necessity of improving public transportation systems. Many of the largest cities in the world have a mixed system comprised of buses and light or metropolitan trains, both of which must serve a continuously rising demand. One option for serving this huge demand is to expand the infrastructure; however, there is a practical limit in such a strategy particularly for train systems. When it is not possible to continue expanding the infrastructure in terms of trains and stations, the only feasible solution for serving the huge demand with an appropriate quality of service is to improve the train's management system. This can be done using offline approaches, such as pre-programming schedules; or online approaches which employ system status information obtained online during operation, including trains' positions, number of passengers at stations and on trains, and eventual disturbances affecting the system's normal operation. Both strategies rely upon a detailed knowledge about effects that changes in operating policy have on the process, obtained from different sources such as operators' experience, historical data, and simulation tools, among others.

In Santiago de Chile, the inauguration of the new Urban Transportation Plan has posed a major challenge for the city's metro rail system: in a very short period of time, the number of people it carries has nearly doubled. According to the Plan,

* Corresponding author. Tel.: +56 2 354 42 81; fax: +56 2 354 25 63.
E-mail address: aciprian@ing.puc.cl (A. Cipriano).

1569-190X/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.simpat.2010.07.012

metro is supposed to be the system's articulated backbone due to the way its routes are structured, its technological integration, and its efficiency in transporting large numbers of passengers. This heavy increase in the number of passengers led to a decline in the quality of service that must be reversed. Several operating strategies have been studied and introduced: infrastructure development, acquisition of new trains, redistribution of the fleet, increase in supply, and informational campaigns to promote a proper use of the service. One of the conclusions gained from this experience is that advantage must be taken of available technology, like simulation tools, in designing and testing new strategies.

1.2. Related work

The introduction of a new planning or control system, be it on or off line, requires prior testing that, at least in the initial stage, is not easy to carry out in the transportation system itself because of the costs involved and the effects on users. The usual solution in these cases involves conducting simulations, and in fact there are many simulation platforms available for these types of systems. One of these is OpenTrack, which is very useful for analyzing the effects of installing new infrastructure, establishing schedules, experimenting with different signal systems, and analyzing the effects of failures [1]. OpenTrack is based on a microscopic model which simulates rail system operations using user-defined trains, infrastructure, and timetables. The software uses a mixed discrete–continuous simulation process and object-oriented programming. The purpose of OpenTrack is to provide a microscopic platform for railroad simulation; therefore, is not suitable for analyzing strategies which consider passengers. Another alternative, which focuses more on learning, is Bahn [2], a shareware program used for designing and testing train or streetcar transportation networks. Although Bahn is able to simulate complex railway systems it is not designed to test operating strategies. RailSys [3], yet another software system, integrates a timetable and infrastructure manager with synchronous microscopic simulation and automatic dispatching. RailSys has been successfully applied in timetable construction, infrastructure planning, and planning of logistic for large scale projects. However, it is not possible to include passengers in the analysis. [4] presents an alternative capable of single-train traction calculation, multitrain simulation and timetable assessment. Other simulators have been proposed that focus on more specific issues, like for simulating inter-modal cargo and passenger transportation terminals [5], or for visually modeling and simulating rail services [6]. However, very few simulators make it easy to design and evaluate online control strategies. Many of the issues involved are discussed in [7]. Examples of these platforms are the Quadstone Paramics [8] and PTV Vissim micro-simulators [9], which are highly complex and costly and do not include specific support for metros. While these platforms are useful for definitively validating the algorithms that have been created, they are not at all practical for carrying out rapid testing during the development phase. One alternative is presented in [10], whose software system supports the generation of simulation codes and is able to automatically define the skeleton of a code. The software allows the user to define the system as a whole by using a relational scheme. It also allows describing the entities as a hierarchy of classes of objects. As case study timetable generation and a control problem for an underground railway network are presented. Although the simulation strategy is clearly stated, neither the model on which the simulation is based, nor the manipulated and measured variables are clear.

1.3. Paper objectives and organization

This article presents an event-driven, fast, and easy-to-use dynamic simulator for multi-line metro systems. The simulator allows users to simulate complex metro systems having as configuration inputs: the time-variant passengers' arrival rate for each station, the origin–destination matrix, the number of trains per line, and parameters defining trains' movement dynamic behavior, trains' capacity, and crowd behavior. The simulator also allows manipulating during the simulations: the train speed along routes, the holding time at stations, and the dispatch time for trains from terminal stations; using control algorithms programmed in Matlab, a popular software package that is used for numeric calculations. The simulator is programmed in the C# language [11,12], which has the advantage of being easy to use, providing good support for object-oriented programming and making it easy to develop graphic interfaces. The paper is organized as follows: Section 2 describes the model of the metro system. Section 3 presents the programming and implementation of the simulator. Section 4 illustrates the application of the simulator in studying two different operating strategies on the Santiago de Chile multi-line metropolitan rail network. Finally, in Section 5 conclusions are drawn and future work is proposed.

2. Modeling

In this section a general stochastic model for the whole metro system is developed. The system is characterized using three main entities: passengers, trains, and stations. For modeling purposes the set of trains M and the set of stations S are broken down into subsets M_i, and S_i, respectively, each of them associated to a line $i \in I$. Subsets of trains and stations form a complete partition of the universes, which means that the intersection between subsets is empty and the union is the universe. For modeling purposes trains are numbered from $m = 1$ to $m = \#(M)$ while stations are indexed consecutively with 0 being the first station and $\#(S_i) - 1$ the last in line i. Then, each station is completely defined by the pair (s,i).

Each bi-directional line has the structure shown in Fig. 1, half the stations belong to each track, with the terminal stations joined by track segments that represent the turnarounds which allow the trains to change from one track to the other. The only care that must be taken is that origin–destination matrices do not allow passengers to travel from one track to the other.

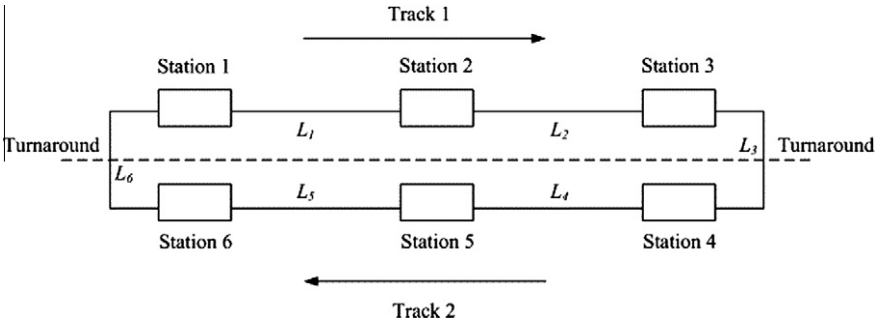


Fig. 1. Structure of one line.

Even though transfer stations belong to more than one line, in the model they are treated as independent stations and transferring effect is addressed as an extra arrival of passengers; thus, for modeling purposes, passengers can only travel between stations belonging to the same line. Once passengers arrive at a transfer station, they are distributed according to the origin–destination matrix of the arriving line.

Passengers arrive at stations according to a composite Poisson distribution with time-variable rates, meaning that they arrive in variable-size groups separated by exponentially-distributed intervals of time. Arrivals are defined by the pair (λ_g, λ_l), where λ_g is the mean size of the group, and λ_l is the group rate, thus, the equivalent passengers' arrival rate is: $\lambda = \lambda_g \cdot \lambda_l$. In order to consider the variability of metro systems demand throughout the day, which is related to rush and lull periods, the model structure allows that the values of the arrival rates change at discrete time instants, yielding a time-dependent rate $\lambda(t)$; in sake of simplicity the time index is omitted in the following.

The proposed model describes the behavior of multi-line metro systems using the parameters presented in Table 1 and the state variables defined in Table 2.

The model is based on events related to arrivals of trains and passengers at stations. When an event takes place the system's state variables are updated in order to obtain the state of the system after the new event. What follows is a description of the equations used.

Given a set of initial values for the states variables, the instant at which each train arrives at the next station is calculated assuming constant train speed over the line segments between stations:

$$tna_m(k) = ta_m(k-1) + td_m(k-1) + \frac{L_{(sa_m(k-1),i)}(k-1)}{v}, \quad \forall m \in M,\ i|m \in M_i \tag{1}$$

In the same form, the instant at which passengers arrive at a station is calculated by generating realizations of the stochastic process:

$$tna_{(s,i)}(k) = ta_{(s,i)}(k-1) + \eta \sim exp(\lambda_{(s,i)}), \quad \forall s \in S_i,\ \forall i \in I \tag{2}$$

Comparing these values determines the train $m^*(k)$ that arrives first at a station and triggers the next train event, or the station $(s^*,i^*)(k)$ at which passengers arrive and triggers the next passenger event, and also establishes the physical time $t(k)$ at which this occurs:

$$t(k) = \min_{m \in M, s \in S_i, i \in I} (tna_m(k), tna_{(s,i)}(k)) \tag{3}$$

If there was a train event, the actualization procedure is as follows:

$$m^*(k) = arg\min_{m \in M}(tna_m(k)) \tag{4}$$

$$ta_{m^*(k)}(k) = t(k) \tag{5}$$

$$i^*(k) = i|m^*(k) \in M_i \tag{6}$$

Since the triggering train has just arrived at a new station (s,i), its station index value must be increased:

$$sa_{m^*(k)}(k) = sa_{m^*(k)}(k-1) + 1(mod\#(S_i)) \tag{7}$$

The total passenger waiting time in person-hours is given by:

$$tw_{(s,i)}(k) = (t(k) - t(k-1)) \sum_{s1 \in S_i} ps_{(s,i),s1}(k-1), \quad \forall s \in S_i,\ \forall i \in I \tag{8}$$

The total number of passengers aboard the train at the moment it arrives is given by:

$$pmt^*(k) = \sum_{s \in S_{i^*(k)}} pm_{m^*(k),(s,i^*(k))}(k-1) \tag{9}$$

Table 1
Model parameters.

C:	Train capacity
$\lambda_{(s,i)}$:	Passenger arrival rate at station (s,i)
α, β, γ, δ:	Coefficients determining train boarding and deboarding times.
$M_{(s1,i1),(s2,i2)}$:	Origin–destination matrix elements indicating the proportion of passengers boarding at station $(s1,i1)$ whose destination is station $(s2,i2)$
$L_{(s,i)}$:	Length of line segment following station (s,i)
v:	Train speed between stations.

Then, the total travel time is given by:

$$tt_{m^*(k)}(k) = pmt^*(k)\frac{L_{(sa_{m^*(k)}(k-1),i^*(k))}(k-1)}{v} \tag{10}$$

The number of passengers deboarding the triggering train is:

$$pd(k) = pm_{m^*(k),(sa_{m^*(k)},i^*(k))}(k-1) \tag{11}$$

We then derive the available capacity and the passengers who will board this train following the FIFO principle:

$$c^*(k) = C - (pmt^*(k) - pd(k)) \tag{12}$$

$$pb_{s1}(k) = ps'_{(sa_{m^*(k)},i^*(k)),s1}(k), \quad \forall s1 \in S_{i^*(k)} \tag{13}$$

Where $ps'_{(sa_{m^*(k)},i^*(k)),s1}(k)$ are the values for $ps_{(sa_{m^*(k)},i^*(k)),s1}(k-1)$ truncated to the available capacity c^* using the FIFO principle.

Passengers getting off at the current station must be deducted from the total for the train, while those getting on must be distributed among the stations they are traveling to:

$$pm_{m^*(k),(s,i^*(k))}(k) = \begin{cases} 0 & \text{if} \quad s = sa_{m^*(k)}(k) \\ pm_{m^*(k),(s,i^*(k))}(k-1) + pb_s(k) & \text{if} \quad s \neq sa_{m^*(k)}(k) \end{cases}, \quad \forall s \in S_{i^*(k)} \tag{14}$$

The number of passengers at the station is updated by deducting those who got on:

$$ps_{(sa_{m^*(k)}(k),i^*(k)),s1}(k) = ps_{(sa_{m^*(k)}(k),i^*(k)),s1}(k-1) - pb_{s1}(k), \quad \forall s1 \in S_{i^*(k)} \tag{15}$$

When a train arrives at a station, it must stay there for a minimum amount of time (dwell time). Dwell time is determined using the model described in [13], which estimates the minimum time required for passengers to board and deboard as a linear function of boarding passengers, deboarding passengers, and the total number of passengers who are at the station and on the train, regardless of whether they board or deboard:

$$T = \alpha + \beta pb(k) + \gamma pd(k) + \delta\left(\sum_{s1 \in S_{i^*(k)}} ps_{(sa_{m^*(k)}(k),i^*(k)),s1}(k-1) + pmt^*(k)\right) \tag{16}$$

Then, the time that a train will stay at a given station is:

$$td_m(k) = \max(T, h_{m^*(k),(sa_{m^*(k)}(k),i^*(k))}(k)) \tag{17}$$

where $h_{m^*(k),(sa_{m^*(k)}(k),i^*(k))}(k)$ is the holding action applied to train $m^*(k)$ at station $(sa_{m^*(k)}(k), i^*(k))$. All the non-updated variables, keep their values unchanged.

If there was a passenger event, the actualization procedure is as follows:

$$(s^*, i^*)(k) = arg\min_{s \in S_i, i \in I}(tna_{(s,i)}(k)) \tag{18}$$

$$ta_{(s^*,i^*)(k)}(k) = t(k) \tag{19}$$

The total passenger waiting time is updated:

$$tw_{(s,i)}(k) = (t(k) - t(k-1))\sum_{s1 \in S_i} ps_{(s,i),s1}(k-1), \quad \forall s \in S_i, \ \forall i \in I \tag{20}$$

The arrivals $psa'_{(s^*,i^*),s1}(k)$ are determined by considering the group size and the origin–destination matrix. For each passenger, his destination is obtained randomly by generating a process whose probability distribution is proportional to the origin–destination matrix.

Then, the number of passengers is updated:

$$ps_{(s^*,i^*)(k),s1}(k) = ps_{(s^*,i^*),s1}(k-1) + psa'_{(s^*,i^*),s1}(k), \quad \forall s1 \in S_{i^*} \tag{21}$$

All the non-updated variables, keep their values unchanged.

Table 2
Model state variables.

$t(k)$	Instant at which event k occurs
$sa_m(k)$	Index indicating station in which train m is located at time of event k
$td_m(k)$	Passenger boarding and deboarding time at station in which train m is located at time of step k
$ps_{(s,i),s1}(k)$	Passengers on station (s,i) whose destination is station $(s1,i)$ at time of event k
$pm_{m,(s,i)}(k)$	Passengers carried by train m traveling to station (s,i) at time of event k
$ta_m(k)$	Moment of arrival of train m at the station in which it is located at time of event k
$tw_{(s,i)}(k)$	Total passenger wait time between events $k-1$ and k in station (s,i)
$tt_m(k)$	Total travel time of passengers between events $k-1$ and k in train m
$ta_{(s,i)}(k)$	Moment of arrival of passengers at the station (s,i) at time of event k

3. Simulator programming and implementation

3.1. General design

The software SimDimMetro has been designed to simulate the global operation of a metropolitan rail system with a proper level of detail. This makes it possible to conduct a satisfactory initial analysis of the effects of strategies like increasing the number of trains in operation, injecting trains at different points along the line, etc., when faced with different conditions of operation and demand, and for any initial system configuration. As outputs, a series of important indicators can be obtained for system planning the most important of which are waiting times, travel times, and stations and trains occupation densities. The software also provides a description of the simulation's total development for its subsequent analysis. The simulator allows the user to fully define the fleet's management strategy, using any desired degree of information available. This can range from sticking to schedules planned beforehand, to complex predictive control techniques [14], with intermediate alternatives like, for example, heuristics that only use local information from the station and/or train under analysis.

3.2. Programming paradigms

In order to implement the model, object-oriented programming [11,12] was used. This is a very popular programming paradigm that consists of defining classes of objects specifically designed to carry out certain functions. Each class defines a set of member data that represents all the information needed to define the state of an object from the corresponding class. It also implements a set of member functions, or methods, that determine the object's behavior, since they can be called on by other parts of the program to act on member data. This makes it possible to modularize the code and allows the abstract objects to constitute a suitable representation of real elements, like metro stations, trains, etc., that share a common class but of which there are multiple instances or objects. Object-oriented programming has been successfully used on simulating complex systems as power generating plants [15]. For the simulation, an event-driven scheme was chosen, in contrast to the continuous simulation paradigm [16]. This means that, instead of integrating dynamic differential or difference equations that define the system's behavior at each moment in time, the choice of the time interval between two steps of calculation is based on the behavior changes of the process and no longer constant [17]; thus, the processing is only conducted at irregular moments of discrete time. At each of these moments, or steps, the system's state variables are updated and the time at which the next step should occur is calculated. What may happen between steps is not taken into account. The advantage of this is to significantly reduce the processing capacity requirements, although details about the process are naturally lost. The use of this paradigm is justified in the fact that all the major events in this application (trains' station arrivals and departures, arrival of passengers, etc.) happen at well-defined moments, and what happens between two events, like for example the exact way in which trains move along the tracks, does not greatly affect operating decisions.

3.3. Characteristics of the simulation

The software can simultaneously simulate any number of routes or lines that are connected by transfer stations. What follows is a description of the diverse aspects of the simulator. The lines are bi-directional and can contain any number of stations, including terminal stations at each end. The time in which trains move along lines and stop at stations depends as much on passenger occupancy as it does on the control strategy used. The speed can be changed at discrete moments by control actions, which means that there may be trains with different velocities. If a train arrives at a station where another train is stopped, it switches to waiting mode until the station is clear, and then moves forward into the station. A similar procedure is followed if one train catches up to another one along the line. There are maneuvering areas at the ends of each line, which have space to store a number of trains as defined by the user. The control determines when to dispatch the train that is first in line in the maneuvering area. In certain sections of the line between stations, the user can define lateral tracks where trains can temporarily park. When a train is dispatched, the control system may determine that instead of functioning normally, it should skip stations and detour to lateral tracks where it will stay parked until the control allows it to resume normal operation. Lateral tracks make it possible to have trains waiting, and when certain parts of the line get overly con-

gested, the train quickly moves to the congested part of the line without having to travel along the rest of the line. Passengers arrival rates can be discretely varied throughout the simulation if the user so desires, as the user may specify different values for different intervals of time. The parameters defining congestion effect caused by an excessive number of passengers are parameters defined by the user. The destination station of passengers who board the train is determined randomly, based on probabilities defined in the origin–destination matrix. When a train arrives at a transfer station, a set proportion of passengers who deboard there transfer to the other line. Not all these passengers take the same amount of time to make the transfer, and a delay effect caused by bottlenecks is taken into account if there is an excess of passengers. Travel times and dwell times can incorporate a random component whose parameters are set by the user. The number of trains running and their initial position are manually specified by the user, who can situate them initially at any station, section along the line, branch, or terminal; indicating values like the time needed to arrive at the current station, or the initial position along a section of line, allowing the software to decide how the trains should behave when the simulation begins. Fig. 2 shows a schematic description of the simulation process.

3.4. Real time control

The simulator can operate in an open-loop or closed-loop mode. In the open-loop mode, all trains have a set speed, stop for the predetermined holding or stop times at all stations, and when arriving at terminals, are dispatched from them at regular intervals. All the parameters that affect these actions can be set by the user. In the closed-loop mode, the simulator allows the user to define control functions that, depending on the desired objectives, may include real-time information about the operation. These functions must be programmed in Matlab using .m files, respecting the entry and exit variables defined by the simulator. The simulator accesses Matlab services (and through these, the function written by the user) through the Dynamic Link Libraries (DLLs) provided by Matlab. The control system can currently act on four types of manipulated variables: holding times, velocities along the line, times until the following train is dispatched from terminal stations, and times until trains are dispatched from lateral tracks. The first two manipulated variables are defined for each train, while the third is associated with lines and the last with lateral tracks. If the dispatch from lateral tracks is being controlled, the user can also manipulate which trains will operate normally and which will be sent to one of these lateral tracks. Each time a train triggers an event, meaning that it arrives at or leaves a station, arrives at a terminal station or at a branch, etc., the control system is called upon to update all the manipulated variables. To do so, it gathers information about the status of the entire system, and transfers this information to the Matlab user-defined function in the form of matrix-represented arguments. The control system generates an exit matrix with all the manipulated variables whose values should change. The program immediately makes the needed changes and then continues on with the simulation.

3.5. Operating interfaces and configuration

The main operating interface of SimDimMetro, shown in Fig. 3 for Santiago Metro network, is the core of the simulator where the user can select different interfaces and configuration boxes; it allows the user to save and load configuration

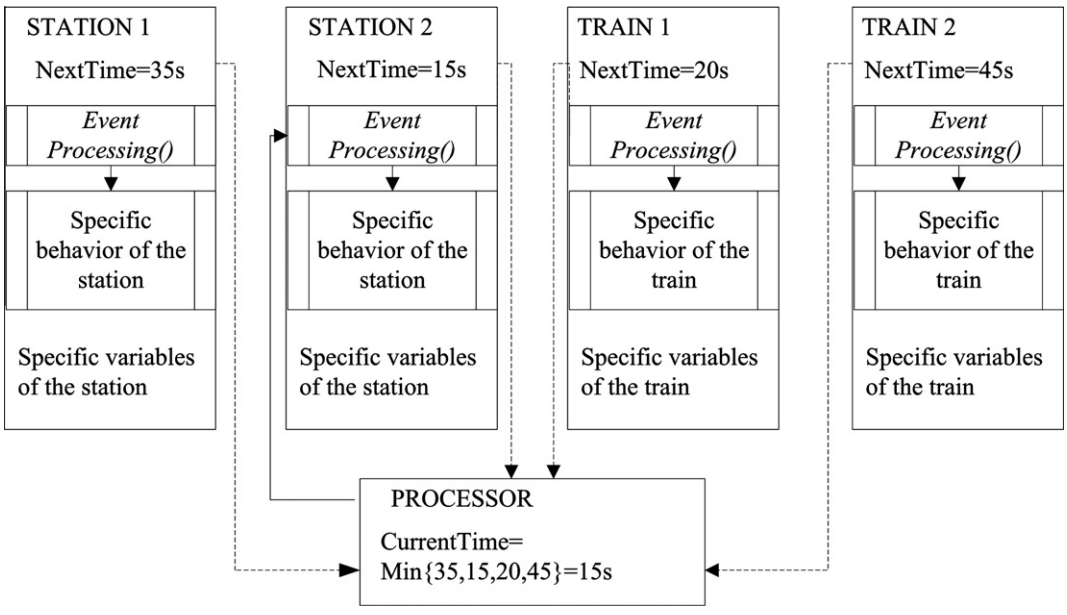


Fig. 2. Description of the simulation process.

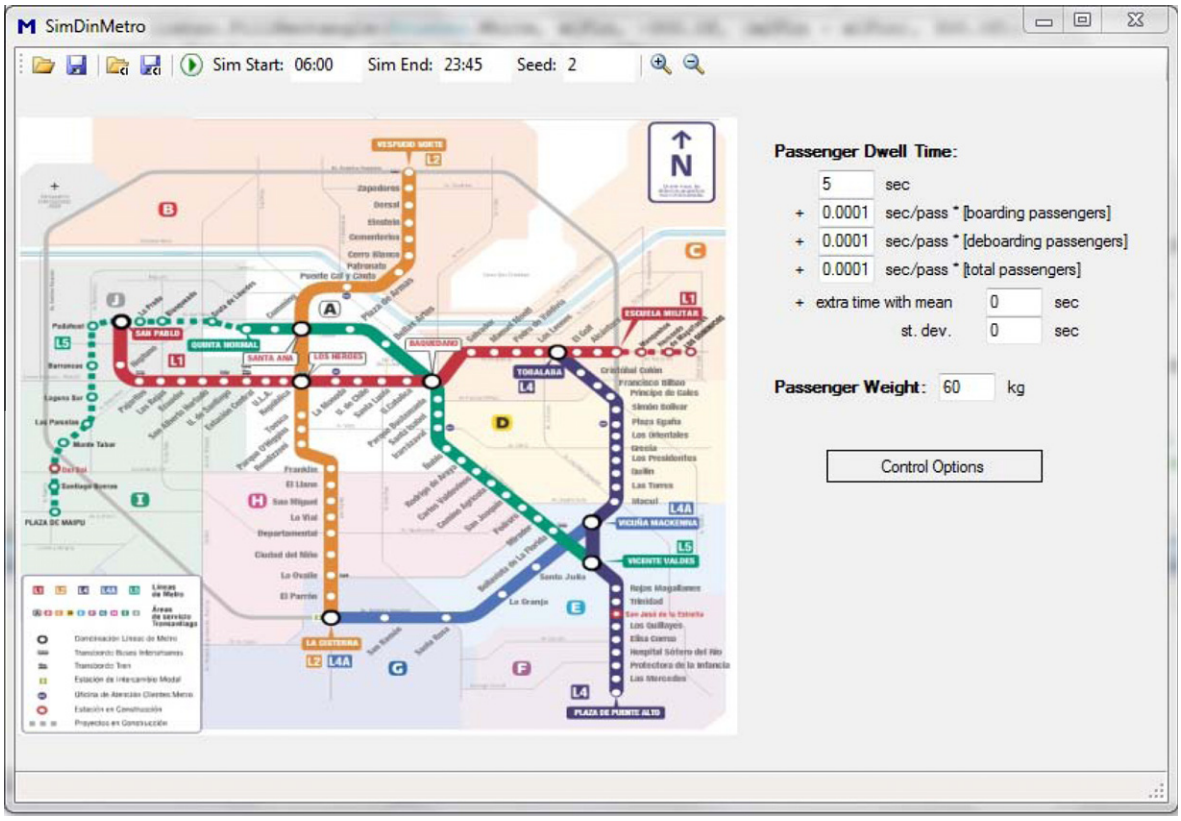
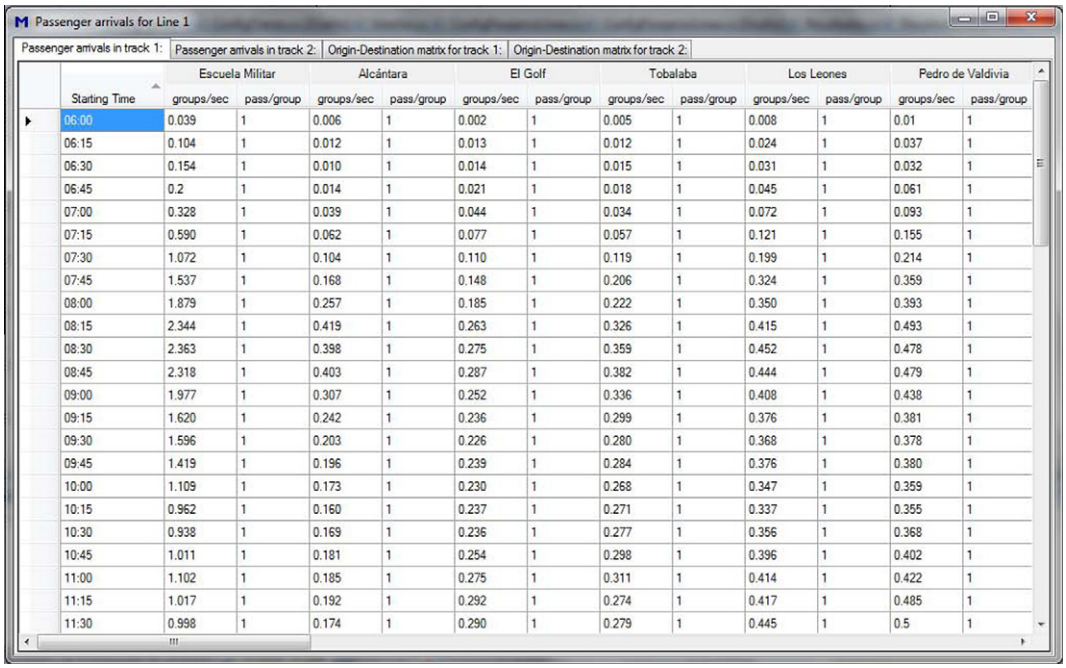


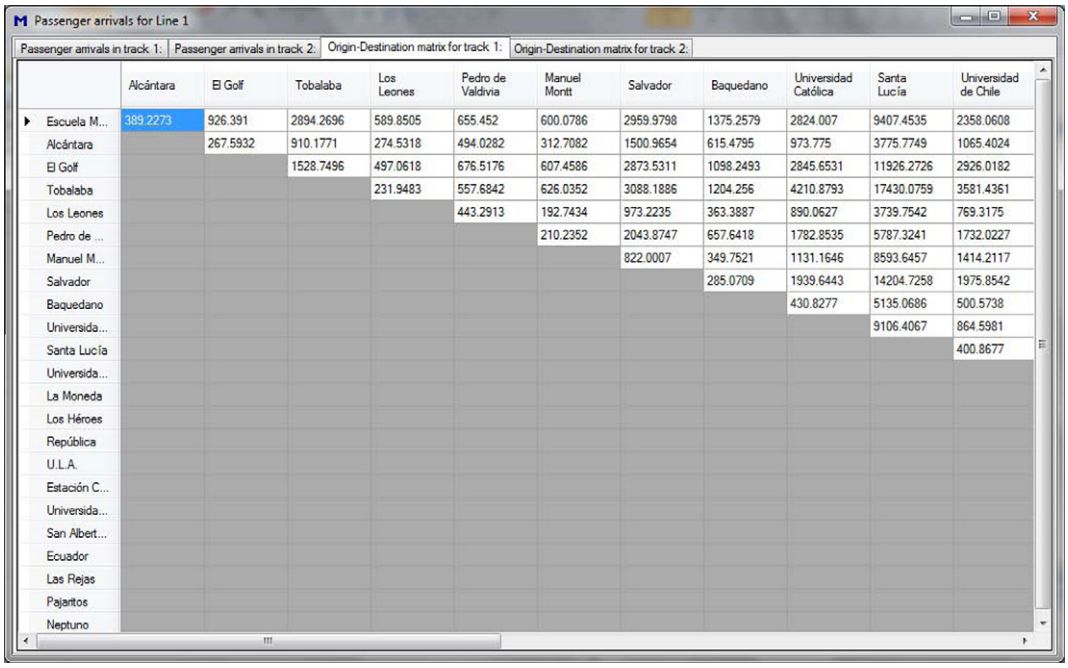
Fig. 3. Main interface of the simulator.



Starting Time	Escuela Militar		Alcántara		El Golf		Tobalaba		Los Leones		Pedro de Valdivia	
	groups/sec	pass/group	groups/sec	pass/group	groups/sec	pass/group	groups/sec	pass/group	groups/sec	pass/group	groups/sec	pass/group
06:00	0.039	1	0.006	1	0.002	1	0.005	1	0.008	1	0.01	1
06:15	0.104	1	0.012	1	0.013	1	0.012	1	0.024	1	0.037	1
06:30	0.154	1	0.010	1	0.014	1	0.015	1	0.031	1	0.032	1
06:45	0.2	1	0.014	1	0.021	1	0.018	1	0.045	1	0.061	1
07:00	0.328	1	0.039	1	0.044	1	0.034	1	0.072	1	0.093	1
07:15	0.590	1	0.062	1	0.077	1	0.057	1	0.121	1	0.155	1
07:30	1.072	1	0.104	1	0.110	1	0.119	1	0.199	1	0.214	1
07:45	1.537	1	0.168	1	0.148	1	0.206	1	0.324	1	0.359	1
08:00	1.879	1	0.257	1	0.185	1	0.222	1	0.350	1	0.393	1
08:15	2.344	1	0.419	1	0.263	1	0.326	1	0.415	1	0.493	1
08:30	2.363	1	0.398	1	0.275	1	0.359	1	0.452	1	0.478	1
08:45	2.318	1	0.403	1	0.287	1	0.382	1	0.444	1	0.479	1
09:00	1.977	1	0.307	1	0.252	1	0.336	1	0.408	1	0.438	1
09:15	1.620	1	0.242	1	0.236	1	0.299	1	0.376	1	0.381	1
09:30	1.596	1	0.203	1	0.226	1	0.280	1	0.368	1	0.378	1
09:45	1.419	1	0.196	1	0.239	1	0.284	1	0.376	1	0.380	1
10:00	1.109	1	0.173	1	0.230	1	0.268	1	0.347	1	0.359	1
10:15	0.962	1	0.160	1	0.237	1	0.271	1	0.337	1	0.355	1
10:30	0.938	1	0.169	1	0.236	1	0.277	1	0.356	1	0.368	1
10:45	1.011	1	0.181	1	0.254	1	0.298	1	0.396	1	0.402	1
11:00	1.102	1	0.185	1	0.275	1	0.311	1	0.414	1	0.422	1
11:15	1.017	1	0.192	1	0.292	1	0.274	1	0.417	1	0.485	1
11:30	0.998	1	0.174	1	0.290	1	0.279	1	0.445	1	0.5	1

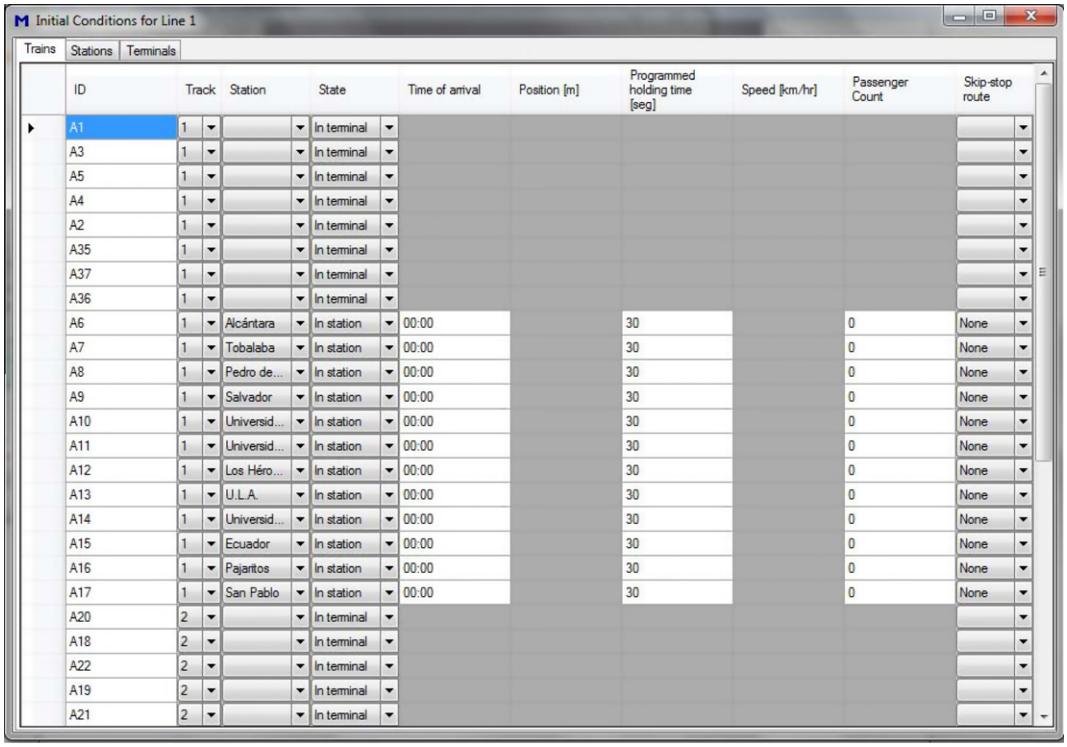
Fig. 4. Passengers' arrival configuration window.

parameters and initial conditions in the form of text files with extension .sis and .cis, respectively. It also allows the user to define the parameters α, β, γ and δ of the dwell time model, and other general configuration parameters like simulation's



	Alcántara	El Golf	Tobalaba	Los Leones	Pedro de Valdivia	Manuel Montt	Salvador	Baquedano	Universidad Católica	Santa Lucía	Universidad de Chile
Escuela M...	389.2273	926.391	2894.2696	589.8505	655.452	600.0786	2959.9798	1375.2579	2824.007	9407.4535	2358.0608
Alcántara		267.5932	910.1771	274.5318	494.0282	312.7082	1500.9654	615.4795	973.775	3775.7749	1065.4024
El Golf			1528.7496	497.0618	676.5176	607.4586	2873.5311	1098.2493	2845.6531	11926.2726	2926.0182
Tobalaba				231.9483	557.6842	626.0352	3088.1886	1204.256	4210.8793	17430.0759	3581.4361
Los Leones					443.2913	192.7434	973.2235	363.3887	890.0627	3739.7542	769.3175
Pedro de ...						210.2352	2043.8747	657.6418	1782.8535	5787.3241	1732.0227
Manuel M...							822.0007	349.7521	1131.1646	8593.6457	1414.2117
Salvador								285.0709	1939.6443	14204.7258	1975.8542
Baquedano									430.8277	5135.0686	500.5738
Universida...										9106.4067	864.5981
Santa Lucía											400.8677
Universida...											
La Moneda											
Los Héroes											
República											
U.L.A.											
Estación C...											
Universida...											
San Albert...											
Ecuador											
Las Rejas											
Pajaritos											
Neptuno											

Fig. 5. Origin–destination matrix configuration window.



	ID	Track	Station	State	Time of arrival	Position [m]	Programmed holding time [seg]	Speed [km/hr]	Passenger Count	Skip-stop route
	A1	1		In terminal						
	A3	1		In terminal						
	A5	1		In terminal						
	A4	1		In terminal						
	A2	1		In terminal						
	A35	1		In terminal						
	A37	1		In terminal						
	A36	1		In terminal						
	A6	1	Alcántara	In station	00:00		30		0	None
	A7	1	Tobalaba	In station	00:00		30		0	None
	A8	1	Pedro de...	In station	00:00		30		0	None
	A9	1	Salvador	In station	00:00		30		0	None
	A10	1	Universid...	In station	00:00		30		0	None
	A11	1	Universid...	In station	00:00		30		0	None
	A12	1	Los Héro...	In station	00:00		30		0	None
	A13	1	U.L.A.	In station	00:00		30		0	None
	A14	1	Universid...	In station	00:00		30		0	None
	A15	1	Ecuador	In station	00:00		30		0	None
	A16	1	Pajaritos	In station	00:00		30		0	None
	A17	1	San Pablo	In station	00:00		30		0	None
	A20	2		In terminal						
	A18	2		In terminal						
	A22	2		In terminal						
	A19	2		In terminal						
	A21	2		In terminal						

Fig. 6. Initial conditions configuration window.

start and finish times, and the random number generating seed. By clicking on different elements on the main screen, the user can access interface windows and dialogue boxes to enter the corresponding configuration parameters and inputs for the simulations.

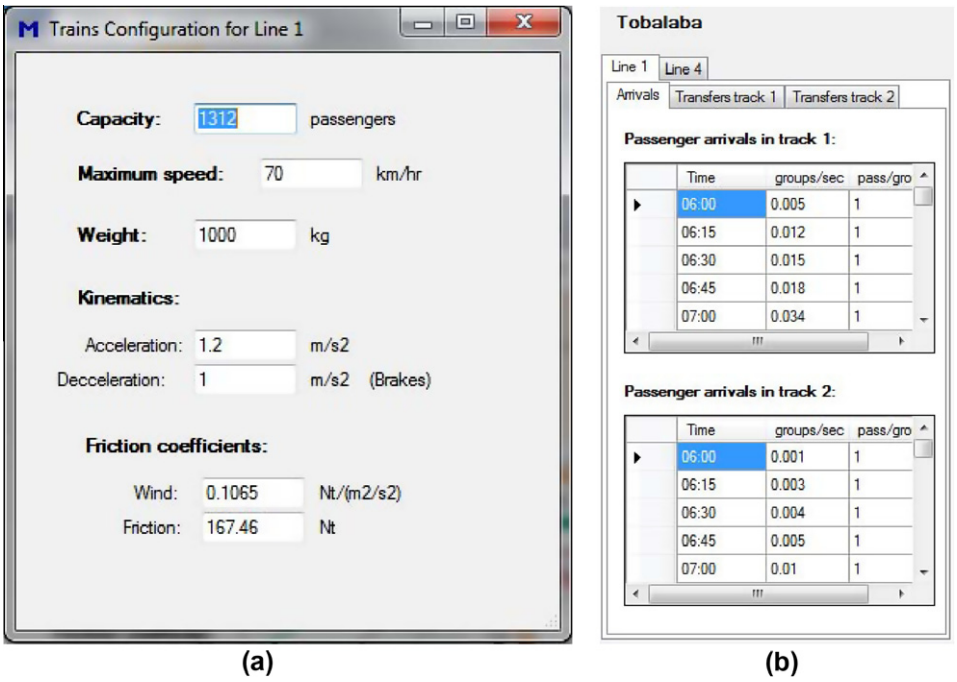


Fig. 7. (a) Train parameters configuration window; (b) Transfer station configuration window.

Fig. 4 shows the interface for defining passengers' arrival rates for the different stations. This interface is accessed by clicking on one line of the metro system, then choosing passengers configuration, and finally selecting the tab labeled as: passenger arrivals. In the interface each arrival interval is defined by the time limits (the same for all stations of the line), the group size and the arrival rate (different for each station).

Fig. 5 shows the interface where the origin–destination matrix for a given line is entered; this interface is accessed by clicking on the corresponding line in the main interface window, then choosing passengers configuration, and finally selecting the tab labeled as: origin–destination matrix. Once the origin–destination matrix is entered, the following step is to input initial conditions. Fig. 6 shows the interface where the initial conditions are defined. This interface is accessed by clicking on the corresponding line in the main interface window and then choosing initial conditions. In this interface the user can configure the number of trains and their initial position which can be at a station, on terminal, or in between two stations. Also the user can configure the initial number of passengers in each train and on each station. Once the initial conditions are entered, the user must define parameters that describe trains' behavior and configure transfer stations. Fig. 7 shows the configuration window for the parameters of trains belonging to a given line (Fig. 7(a)), and for the parameters of one transfer station (Fig. 7(b)). Trains' parameters required are: capacity, maximum speed, weight, and cinematic characteristics. These parameters yield for all the trains in a given line. For transfer stations the user must enter the proportion of passengers who transfer from one line to the other, and the parameters defining the time required for doing the transfer.

Similar boxes can be called up to define the control strategy. Fig. 8 shows the dialogue boxes for open-loop control (Fig. 8(a)) and closed-loop control (Fig. 8(b)). For open-loop control, the user must define the speed, stop time at each station, and the value of dispatch intervals. For closed-loop, or real time, control the user must provide the route of the Matlab file that implements the desired control algorithm; also six user-defined constants can be introduced, if desired, for fast communication between the user and the Matlab control algorithm. The simulation begins by clicking on the play button at the main interface window.

Once the simulation is finished, the user can access statistics about the metro line or station, including: average passenger waiting time (per station or for the entire line), average travel time, average occupancy of stations and trains, average frequency of trains in a station, and total number of trains that must wait to enter a station because it is occupied by another train. Fig. 9 shows the statistics for a given line (Fig. 9(a)) and a given station (Fig. 9(b)). In the results per station interface (Fig. 9(b)) results are presented one by one. The user is able to change the station for which results are being presented by selecting the desired station on the bottom status bar drop menu. Besides statistics, two types of graphic representation are created after the simulation is finished. The first is a distance-time diagram for each line, an example of which is shown in Fig. 10. In this graph, X-axis represents time and Y-axis represents the distance of the entire metro line; each line in the graph represents a train as it moves along the metro line. The simulator also creates a graph of the number of passengers in each station as a function of time, as shown in Fig. 11. In this graph light areas represents the time period when there is no train at the station, and dark areas represents time periods when a train is at the station. Lastly, it is possible to save a register of the

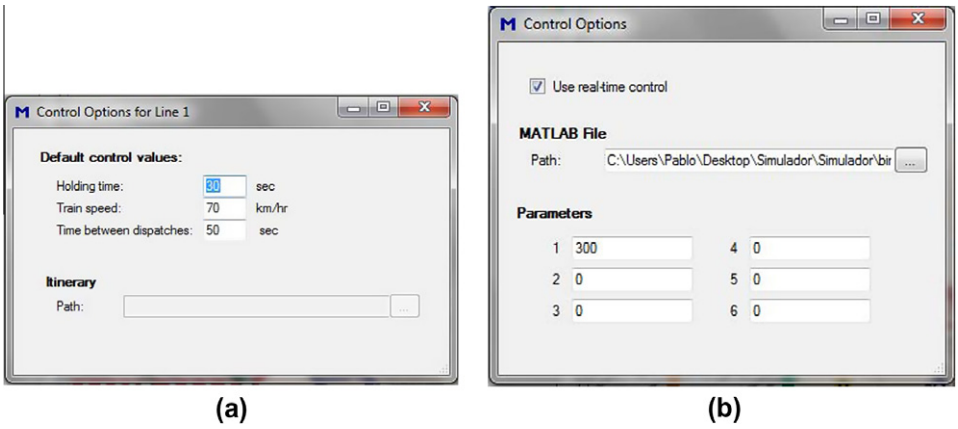


Fig. 8. (a) Configuration window for open-loop operation; (b) Configuration window for real time control (closed-loop) operation.

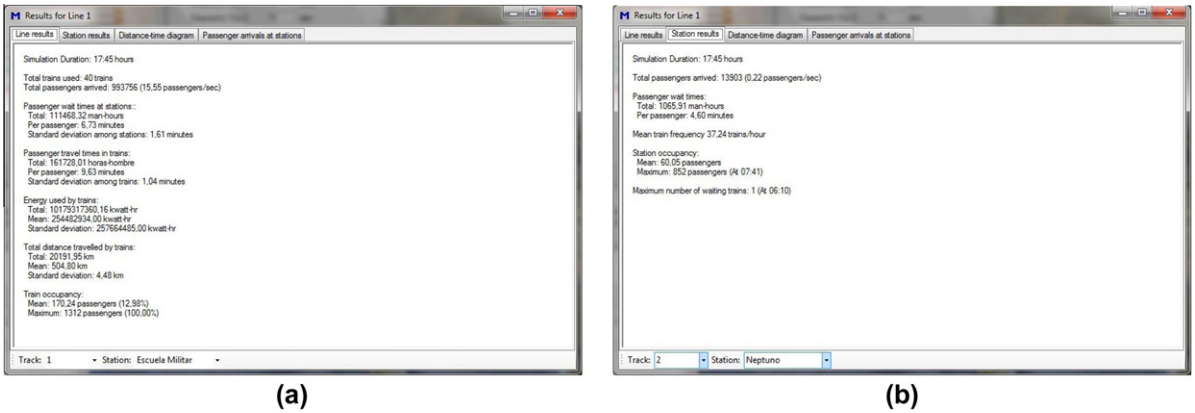


Fig. 9. (a) Simulation results for one line; (b) Simulation results for one station.

simulation's development as a Matlab data file (a .mat file) where the user can check the temporal evolution of all the state variables of the system.

4. Application example

As a case study, simulations of the Santiago de Chile Metro, formed by 5 lines (Line 1, Line 2, Line 4, Line 4A, Line 5), 93 stations, and 7 transfer stations are presented. Santiago de Chile Metro is the most important element of the city transportation system. Metro operates from 6:00 to 23:45 hrs on weekdays (8:00 to 22:30 hrs on weekends), and it transports 2.3 million passengers per day. Due to the importance of Metro several efforts have been done to improve operation in terms of service quality, including the use of state of the art fleet management strategies and a continuous upgrading of the measurement system. Metro measurement system identifies the hourly demand per station based on the use of electronic tickets and pre-paid cards. However, destination stations can not be identified directly from tickets, requiring the use of estimations that in this case are survey-based. Metro conducts surveys continuously during the year and constructs the origin–destination matrices, used on fleet management, based on these surveys.

The simulator was calibrated to simulate the entire network of the Santiago de Chile Metro using real time-variant passenger arrival rates that correspond to a normal business day. Arrival rates of each station, are varied each 15 min during the weekday time period: 6:00 to 23:45. Real origin–destination matrices obtained from surveys and real train and station parameters are also used in calibration. In order to show how the simulator works, a comparison between the performance of two simple real-time control approaches is proposed. Both approaches are based on regularizing the headway of station departures, that is, the time between successive train departures from a given station. A classic result in transportation engineering is that in a system without published schedules (which is generally the case with high-frequency systems such as a metro) and a fixed average frequency (determined by the number of trains) the regularization of headways tends to reduce passenger waiting times [18]. Thus, the performance of the two operating strategies is compared using passengers waiting

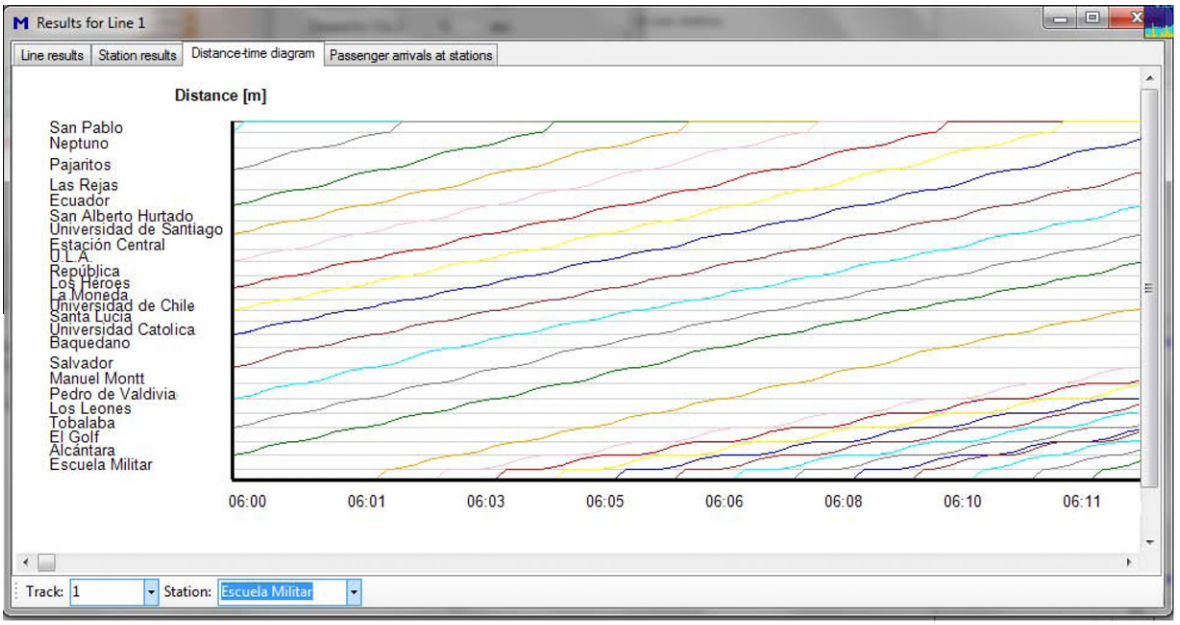


Fig. 10. Time–distance graph for one line.

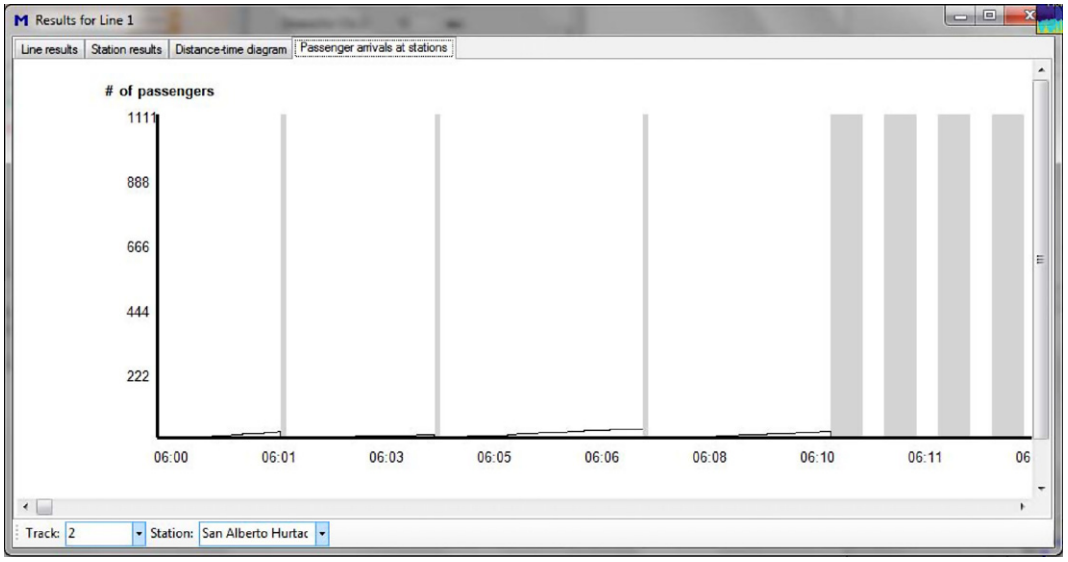


Fig. 11. Passengers' graph for one station.

time as main decision criteria. The first strategy intends to maintain the headway constant, during all the simulated time, at a specific value H_i for each line as shows Table 3. The second strategy also aims to maintain the headway H_i constant, but at a value that depends not only on the line, but also on the hour, as shows Table 3. Both strategies use the following reasoning: every time a train arrives at a station s of line i at a moment t_{arr}, the moment when the previous train departs the same station, $t_{prev}(s_i)$ is obtained from the measurement system. The holding h_{n_i} for the train n in line i is then calculated using the following rules:

$$\text{IF } t_{arr} \geqslant t_{prev}(s_i) + H_i \quad \text{THEN } h_{n_i} = 0 \tag{22}$$

$$\text{IF } t_{arr} < t_{prev}(s_i) + H_i \quad \text{THEN } h_{n_i} = t_{prev}(s_i) + H_i - t_{arr} \tag{23}$$

This means that the control system will try to keep the station headway (the same for all stations belonging to line i) constant by holding the train at the station. It is important to note that if the train arrives very late the holding is zero, so it can

Table 3
Headway per line [s].

	Case 1			Case 2		
		06:00–10:00	10:00–12:00	12:00–14:00	14:00–17:00	17:00–23:45
Line 1	130	120	150	113	150	116
Line 2	175	138	200	180	200	158
Line 4	180	172	185	185	185	168
Line 4A	200	200	200	200	200	200
Line 5	145	120	160	160	160	125

Table 4
Simulation results.

	Line 1	Line 2	Line 4	Line 4A	Line 5
Case 1					
Average passenger waiting time [s]	45.27	69.61	80.40	81.63	59.02
Trains' average occupation [%]	10.72	9.39	5.31	1.63	7.28
Trains' maximum occupation [%]	98.93	100	48.55	16.98	62.25
Case 2					
Average passenger waiting time [s]	45.26	52.27	74.40	81.63	46.57
Trains' average occupation [%]	11.21	11.05	5.46	1.63	8.32
Trains' maximum occupation [%]	91.54	76.26	50.00	16.98	55.77

move through the station as fast as possible, considering that it will need a minimum amount of time for all the passengers to board and deboard.

Both control approaches were tested under normal operating conditions from 06:00 to 23:45 h. The parameters used for the dwell time model were: $\alpha = 5$ s, $\beta = 0.01$ s, $\delta = 0.02$ s, and $\gamma = 0.005$ s. Table 4 shows the simulation results for the two cases. It can be seen that when a variable headway strategy was used (case 2) the average passenger waiting time decreased in the five lines. Also when the system was operated in a variable headway strategy, the trains' average occupation increased and the maximum occupation decreased (except for line 4) suggesting that under a variable headway strategy the system operation is more efficient. Based on simulation results, it can be said that operating using an hourly-dependent headway strategy improves the system performance not only from the user point of view (waiting time), but also from an operational point of view (trains' occupation).

5. Conclusions and future research

5.1. Conclusions

The event-driven simulator for multi-line metro system developed using object-oriented programming is capable of reproducing the behavior of metro systems. It allows the user to calibrate the simulator with the desired time-variant passenger's arrival rates, origin–destination matrices, and train and stations characteristics. As outputs the simulator delivers performance indicators, such as: average passenger waiting time, trains' occupation, passengers' behavior graph for a given station, and time-distance graph for a given line, among others, useful for evaluating operating strategies.

The presented simulator simplifies some details involved in the operation of a metropolitan rail system, especially those related to train dynamic motion along the line, in order to facilitate the programming and reduce computations. These simplifications make it possible to conduct rapid simulations of complex systems, like the Santiago Metro network, while maintaining the level of detail required for evaluating operating strategies which consider the system as a whole.

A simple application regarding a comparison between two holding strategies in the Santiago metropolitan rail network illustrates the capabilities of the simulator in evaluating different operating schemes. Results show that operating using an hourly-dependent holding strategy improves the system performance in terms of passengers' wait time and trains' occupation.

5.2. Future research

Future research includes the use of the simulator in designing and testing advanced control approaches for metro systems by exploiting its ability to connect to Matlab. Also of interest is conducting a study regarding the effects of single- and multi-line service interruptions and the strategies for returning to normal operation.

Future research should also address improving the simulator by adding the option of using detailed train motion models in a continuous time simulation mode to study the optimal design of future stations or metro lines. An interesting additional

development that is being considered for the future is the ability to connect the simulator through an interface to data available online, to create a decision-support system.

Acknowledgments

This research has been funded by CONICYT, ADI-32 Project: "Real Time Intelligent Control for Integrated Transit Systems". We also wish to thank the Santiago Metro Projects Department for its suggestions and comments.

References

[1] A. Nash, D. Hürlimann, Railroad simulation using OpenTrack, in: J. Allan, C.A. Brebbia, R.J. Hill, G. Sciutto, S. Sone (Eds.), Computers in Railways IX, WIT Press, Southampton, 2004, pp. 45–54.
[2] JB BAHN – Simulation of railway and streetcar networks, <http://www.jbss.de/hpg_eng.htm>, 2010 (accessed 05.10).
[3] A. Radtke, D. Hauptmann, Automated planning of timetables in large railway networks using a microscopic data basis and railway simulation technique, in: J. Allan, C.A. Brebbia, R.J. Hill, G. Sciutto, S. Sone (Eds.), Computers in Railways IX, WIT Press, Southampton, 2004, pp. 615–625.
[4] M. Baohua, J. Wenzheng, C. Shaokuan, L. Jianfeng, A computer-aided multi-train simulator for rail traffic, in: Proceedings of the IEEE International Conference on Vehicular Electronics and Safety ICVES, 2007, pp. 1–5.
[5] A.E. Rizzoli, N. Fornara, L.M. Gambardella, A simulation tool for combined rail/road transport in intermodal terminals, Journal of Mathematics and Computers in Simulation 59 (1–3) (2002) 57–71.
[6] V. Čerić, Visual interactive modeling and simulation as a decision support in railway transport logistic operations, Journal of Mathematics and Computers in Simulation 44 (3) (1997) 251–261.
[7] C. Verseegt, A. Verbraeck, Supporting the design of automated transport systems using innovative simulation, in: R. Konings, H. Priemus, P. Nijkamp (Eds.), The Future of Automated Freight Transport: Concepts Design and Implementation, Edward Elgar Publishing, Cheltenham, UK, 2005, pp. 167–181.
[8] Quadstone Paramics 65 Modeller User Guide, Quadstone Paramics Ltd., Edinburgh, UK, 2006.
[9] PTV Vision – Tutorial, VISSIM Basic Network, PTV Planung Transport Verkehr AG, Karlsruhe, Germany, 2008.
[10] M. Paolucci, R. Pesenti, An object-oriented approach to discrete-event simulation applied to underground railway systems, Simulation 72 (6) (1999) 372–383.
[11] B. Stroustrup, The C++ Programming Language, Addison-Wesley, Massachusetts, USA, 1997 (Special edition).
[12] J. Sharp, Microsoft Visual C# 2008 Step by Step, Microsoft Press, 2007.
[13] H. Lin, T.N. Wilson, Dwell time relationships for light rail systems, Transportation Research Record 1361 (1992) 287–295.
[14] E. Camacho, C. Bordons, Model Predictive Control, Springer Verlag, London, 2004.
[15] J. Garrido, A. Zafra, F. Vázquez, Object oriented modelling and simulation of hydropower plants with run-of-river scheme: a new simulation tool, Simulation Modelling Practice and Theory 17 (10) (2009) 1748–1767.
[16] R. Hill, T. Yates, Modelling railway block signalling systems using discrete-event simulation, in: Proceedings of the ASME/IEEE Spring Joint Railroad Conference, 1992, pp. 1–9.
[17] N. Giambiasi, J.C. Carmona, Generalized discrete event abstraction of continuous systems: GDEVS formalism, Simulation Modelling Practice and Theory 14 (1) (2006) 47–70.
[18] C.F. Daganzo, Fundamentals of Transportation and Traffic Operations, Pergamon, Oxford, U.K., 1997.