A Java based web application which addresses urban traffic problems relying on real time collected data.

F. BELLOTTI, A. DE GLORIA, D. GROSSO
Department of Biophysical and Electronic Engineering (DIBE)
University of Genoa
Via Opera Pia 11a 16145 Genoa
ITALY

Abstract: - *Abstract:* - In this paper we present a web application entirely developed in Java. The application deals with the problem of computing the shortest path between a chosen origin-destination pair, in a urban context with multimodal means of transport and time dependent traffic conditions. The city is modeled as an adirected graph, where each node represents a strategic point of the city. The overall application consists of a graphic interface embedded in an html page and of a server program. Through the interface the user can interact with a remote server that computes the smallest time path for the user selected couple of departure and arrival nodes. Particular attention is spent to provide a realistic model of the multimodal transportation system. Finally we provide an estimation of our system's performance on different operative platforms. IMACS/IEEE CSCC'99 Proceedings, Pages: 1121-1127

Key-Words: - Web, Java, traffic, multimodal transport, shortest path, personal communication systems, real time.

1. Introduction

The traffic problem is one of the main unresolved issues that threaten the quality of life in modern cities. A bad utilization of the public means of transport, even more than specific shortcomings in the service, is a not negligible reason that can explain the massive resort to private means by most of the citizens. This is essentially due to a lack of information on the user part. In fact local transport schedules, especially in wide and complex urban areas, are often diffcult to know or to remember even because of the wide range of services that the transport companies have to provide. Moreover schedules and travel time may vary during the day because of different traffic conditions, that are not always predictable. Since all this mass of information is unlikely to be known by the people, and in particular by tourists, a good solution to the problem could be the utilization by the user of a communication device through which he or she can connect to a centralised server that computes the best transport solution consulting an accurately real time updated database.

This paper presents a web-based implemetation of an algorithm aimed at finding the minimum path, in terms of elapsed time, between two points in a town. As a proof of concept we applied this solution to the city of Genoa (Fig 1), where traffic is a critical problem, but the algorithm is obviously generally applicable. At present the public transport net of Genoa consists of capillar bus and railway services and of a short underground (Fig 2). We built a model of the town as a net of interconnected points that represent bus stations, railway stations and other important places of the town. At each node is associated a frequency of departure for the public means of transport, while at each branch is associated the time required to cover it by any possible means of transport. Our solution provides a simulationless computation of the minimum time path connecting the chosen start and arrival points.

The programming language chosen for the overall implementation of the project was Java, principally for its intrinsic features of portability, security code density[1, 2] and robustness, that make it particularly suited to applications that can be deployed on computer networks[3, 4, 5].

In the present specific case, clients can download a visual-appealing, easy to manage applet from the internet on either their desktop or personal communication system or car computer or on small coputers that may be located in special kiosks near bus stops. By means of the applet's graphic user interface, customers can submit to a service provider information about their planned journey through the city (typically: departure and arrival places, time of departure) and get in real time the most suitable path to go to destination. Customers can choose which means of transport to use from a set of predifined means including train, car, taxi, bus and metro. The possibility of going on foot is also considered.

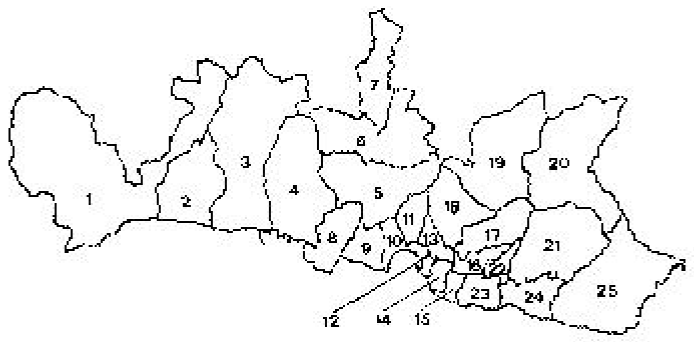


Figure 1 A schematic representation of the central zones of the city of Genoa. The famous old harbour is located by the zones numbered from 9 to 14.

Figure 2 Bus and train of the Genoa's public transportation system.

Real time information about traffic, viability and train circulation is gathered from different on the field dislocated information sources and elaborated by a server program, written in Java language. The program is an extended implementation of the Dijkstra's shortest path algorithm [6]. The extension primarily consists in keeping into account the possibility of using different means of transport during the journey. Connections between different means, also of the same kind (i.e. bus-bus connections), have been considered and carefully modeled at any stations. Further problems and aspects have also been tackled, such as the run-time minimization of the network connections complexity in order to reduce computation time without compromising the quality and the correctness of the resulting information. Information about car park availability is also concerned.

In the absence, at present, of a net of traffic sensors deployed in the city, for the current implementation it has been chosen to follow a completely analytical approach. In particular we use statistically computed mean values for certain variables such as the travelling time between different nodes. Statistics are obtained from on the field information.

2. The Model

The model we built to represent the city of Genoa is a network of 45 interconnected nodes. We split the big Genoa area in 5 geographical zones and we took as nodes the more important places of each zone. The nodes represent either a train station or a strategic point for the traffic or a place where economic, cultural or other important activities take place. Fig. 3 shows a simplified network scheme and the position of some nodes on the territory. The nodes can be connected either by public transport means, such as bus, train and taxi, or through private car or on foot, as shown in Fig 4. Each branch of the graph is assigned a weight that represents the time spent to go through it. For example from the Fig 4 we deduce that it is possible to go from node CO to node BZ in 4 minutes using the train, in 19 minutes using the bus, in 16 minutes using the car and so forth. In addition to the data described so far, that concern with inter-nodes transport, the model receives as input other information describing the modality of change of means of transport within each node. Usually two nodes can be connected by more than one bus or train lines; in this case the model specifies the frequency of each bus and train line, in order to provide users with more precise information about which means to take and how much time to wait at a node to get the next means, as will be more clear from the description of the algorithm. For simplicity, we consider that all the bus lines connecting two adjacent nodes take the same time to reach the destination. However even this constraint, though realistic, can be removed. Each node is also assigned two parameters that indicate the time spent to park the car and the time required to have a taxi ready.

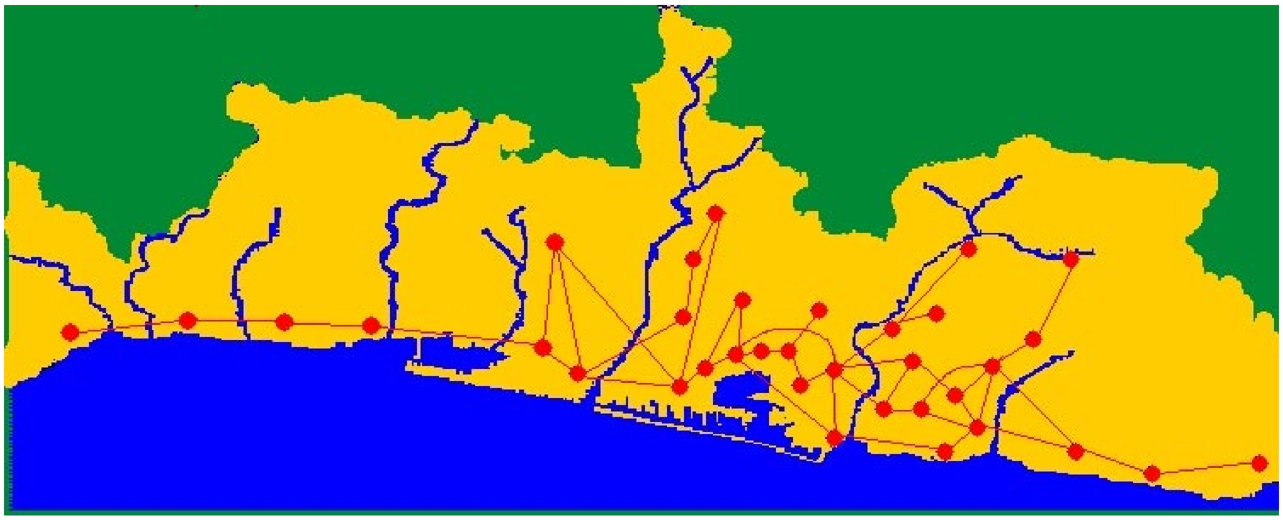


Figure 3 A portion of the network of nodes that models the city

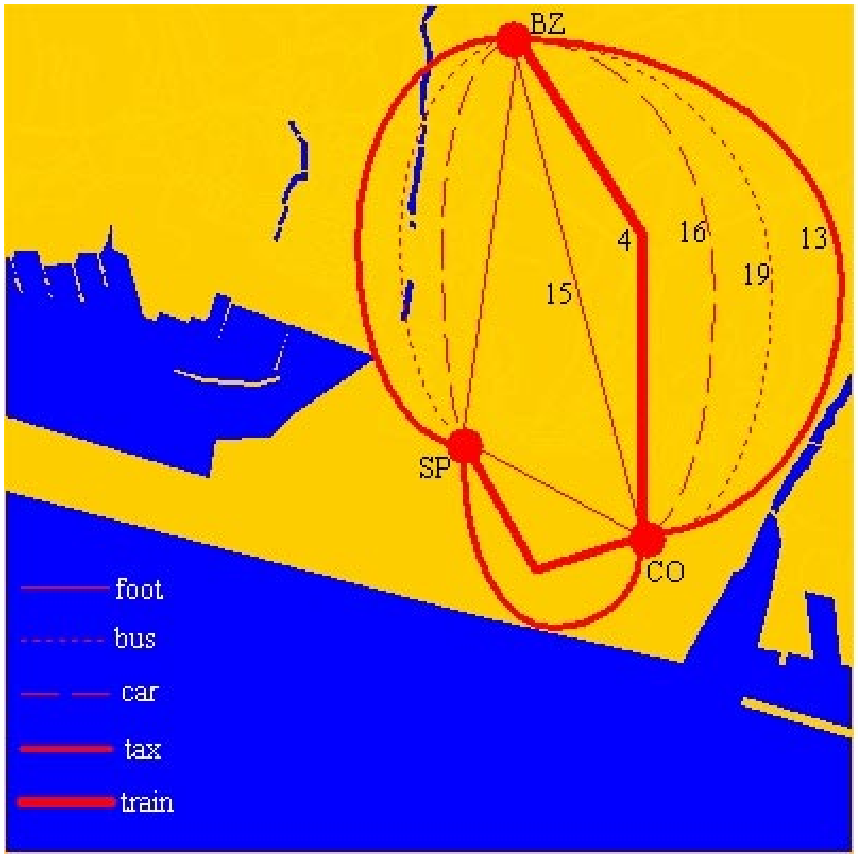


Figure 4 Three nodes connected by different means of transport, and the associated travelling times.

All the values and parameters of the model are taken from the real world. For example, the train frequency and the travel duration are taken from the official timetables of the national railway and for the bus we consider the data supplied by the local transport company[7]. As it happens in the real world the schedules for all the transport means are variable in function of the considered day and hour. And the same is true for the duration of the car travel. At present we do not take into account the financial costs connected with the travel (i.e. the cost of taxi, fuel, bus or train fare…). A precise model at this regard, though, would not be difficult to elaborate. Moreover, as long as only public transport means are concerned, the overall cost is easily computable since there is a 90-minute validity unified ticket that allows people to move anywhere in the city using buses and trains at the cost of £1500 (US$ 0.9).

In order to speed up the computation of the algorithm we consider the overall area as the union of 5 zones of the town. Some nodes, in particular the most important ones, can belong to two neighbour zones and are included in the graphs of both. When the user select the start and the arrival stations our system dynamically builds a reduced graph that consists of the smallest network, according to the 5 zones model, that contains the two stations. The subdivision is suggested by the analysis of the traffic flows that are mainly directed inward or outward the centre, respectively from or to more peripheral areas. Moreover, this reduction is particularly appealing in the case of Genoa, that has a prominent latitudinal configuration and thus it quite easily lends itself to a subdivision in partial areas. At this point the algorithm can begin its work on this partial network.

3. The Program

The overall service we have developed consists of two different applications, both written in Java. One is on the client side and is the applet which can be downloaded through the Internet by the user at home or in his/her car or on the airplane arriving at the city. The applet is the interface that collects data representing the user's request. The other application is a server program which is triggered by user's data, executes the algorithm that finds the shortest path between the departure and destination nodes and sends the results back to the user.

3.1 The Applet

The user interface is implemented by means of an interactive Java applet (Fig 5). The client is invited to choose the departure and arrival nodes between a list of 45 fixed nodes. The user can also specify which kind of means of transport he or she intends to take. From another point of view, he can select which means (e.g. taxi, car) not to use. Other information concerns the forseen hour of departure, because the traffic conditions and the schedules obviously heavily depend on it. This is the basic model of interaction with the user. More complex models of interface may involve the enabilitation/disabilitation of features such as the change-times at nodes, as we will furhter describe. Data so far collected is sent to the server via socket. Then the applet waits for the response from the server and displays the final information showing the overall cost, in terms of time, of the travel and detailing for each covered node which means to use, how much time to spend and eventually, in the case of bus or train connection, which lines to take and the final destination of each line [1] (Fig 6). It is worth to note that in the last case, not all the lines connecting two nodes are reported, but only those that are effectively usefull to user, as will be explained in the algorithmic section. More attractive and pleasant interface, possibly invilving the visualization of the path on a map, may obviously be developed in the future.

3.2 The Server side program

The server program is triggered by user's data. It initially builds the dynamic graph, in order to reduce the computational effort, as described in the previous section. Then it executes an ad hoc specialized version of the shortest path algorithm.

3.2.1 The shortest path algorithm

The most efficient algorithm for the solution of the s-a (start-arrival) shortest path problem was given initially by Dijkstra [8, 9]. The method described below applies to adirected cyclic trees with the proviso that no negative cost circuits exist and it consists in assigning temporary labels to the vertices of the graph. Each label records the expected cost to reach the corresponding vertex from the departure node. These labels are then updated by an iterative procedure that converges in at most N-1 iterations, where N is the number of nodes in the graph. A simple, not optimized, version of the Dijkstra algorithm is sketched below:

Let $J^k(x)$ be the label on vertex x at the k-th iteration of the algorithm.

Initialization

$J^0(s)=0$.

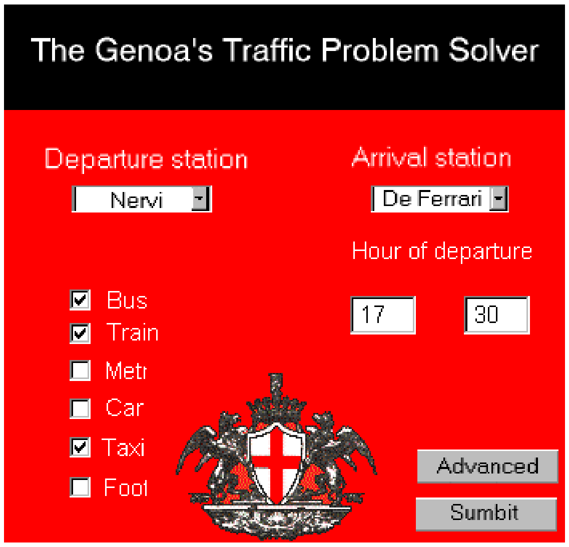
$J^0(x)$ is set to Inf. (practically, a very large numeric value, as compared to the other partial costs) for all the other x's.

Updating of label

For each k-th iteration, with k in[1,N-1], for each node x the updated cost is computed.as shown in (1);

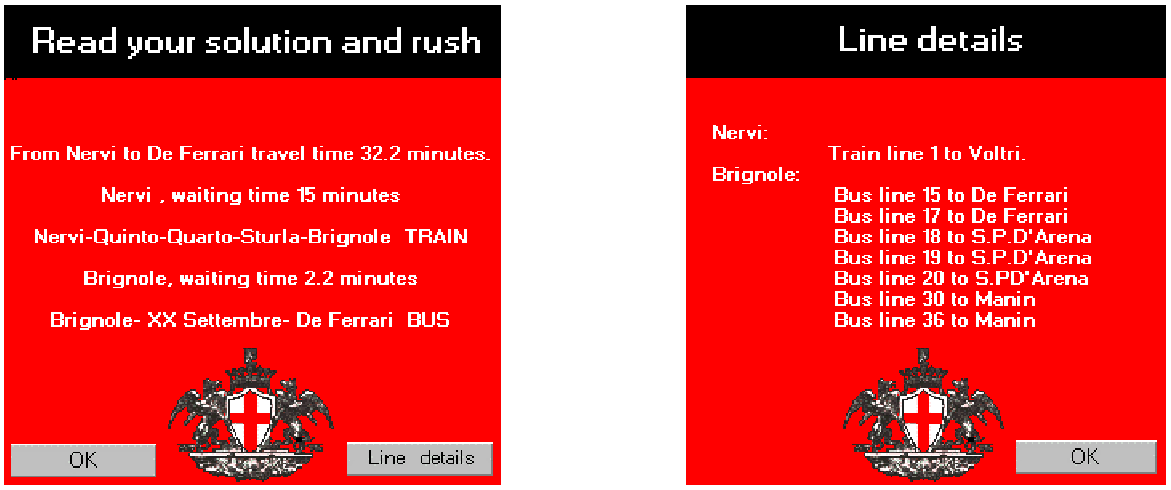
$$J^{k}(x) = \min_{v \in V(x)} \{C(v,x) + J^{k-1}(v)\} \qquad (1)$$

[1] This features is ideallly suited to tourists that have no knowledge of the city.



Selected: [✓ Bus] [✓ Train] Metr Car [✓ Taxi] Foot

Figure 5 The interface through which the user can get information aout his planned path.



From Nervi to De Ferrari travel time 32.2 minutes.

Nervi , waiting time 15 minutes

Nervi-Quinto-Quarto-Sturla-Brignole TRAIN

Brignole, waiting time 2.2 minutes

Brignole- XX Settembre- De Ferrari BUS

Nervi: Train line 1 to Voltri.

Brignole: Bus line 15 to De Ferrari
Bus line 17 to De Ferrari
Bus line 18 to S.P.D'Arena
Bus line 19 to S.P.D'Arena
Bus line 20 to S.PD'Arena
Bus line 30 to Manin
Bus line 36 to Manin

Figure 6 The output of the program shows the travel time and some details on the adviced means of transport

where V(x) is the set of the neighbourgh nodes of node x. The complexity of the algorithm is O(N^3), where N is the number of nodes in the tree.

3.3 The modified version of the algorithm

The model we have developed involves the utilization of different interchangeable means of transport. This means that multiple labeled branches may connect two adjacent nodes and that at each node the eventual time required to change the means is to be calculated. In order to meet our specifications, Dijkstra's algorithm has been modified in that at each iteration, for each node n, the cost (cost(c, n)) is computed to reach it with each kind of selected means of transport c.
In the end, the best solution for the arrival node a is singled out as:

$$\min_{c \in SMT} \{\cos t(c,a)\} \tag{2}$$

Where SMT is the pool of the selected means of transport.
The overall complexity of the algorithm is O(CN^3), where C is the number of considered means of transport. The core of the algorithm consists in the upadate_cost function, that computes the next_cst value (i.e. the time that it takes the user to arrive to node n with the c mean) :

$$update\_\cos t(c,n) = \min_{v \in V(n)} \{t(c,n,v) + + \min_{c^* \in SMT} \{\cos t(c^*,v) + t_{change}(v,c^*,c,n)\}\} \tag{3}$$

where the outermost minimum is computed on all the nodes that can directly reach node n, while the innermost is computed on all the selected kinds of vectors. T(c,n,v) is the time required to go from n to v with the c means. If n is the arrival node, it must be added a final cost for cars (i.e. the time required to park the car[2]).

The following rule applies to the departure node s:

$$J^0(s,c) = 0 \quad \forall c \in SMT \tag{4}$$

While the initial cost for s is added as the term t_{change}(s, ,c,n) of the equation used to compute the cost to reach a neighbour node n from s. Table 1 shows the initial costs for the different means.

Car	0
Train	Train_waiting_time(s,n)
Bus	Bus_waiting_time(s,n)
Foot	0
Taxi	Taxi_waiting_time(s)
Metro	Metro_waiting_time

Table 1 Initial costs for the different means of transport

Metro_waiting_time is constant for all the stations, because there is only one metro line in Genoa. For simplicity we have assumed that the initial cost for a car is 0, although an advanced interface option can allow user to specify his/her time required to reach the car. On the other hand, for means such as buses and trains that involve the utilization of different lines, a more complex function is required. In this case the waiting_time is the expected value of the probablity that user waits for more than T minutes. In the case of N different lines connecting node s to n, the probability is shown below:

$$P(t > T) = (1 - P_1(t))\,(1 - P_2(t))\ldots(1 - P_N(t)) \tag{5}$$

Where $P_i(t)=f_i t$ is the probablity that a bus of line i arrives within t minutes and f_i is the frequency of the i-th line[3].

Similar considerations apply to intermediate nodes. The change time at a station v is the sum of a fixed term[4] and a probabilistic term dependent on the availability of the means and on the schedules. Table 2 describes all the possible intra-node change times combinations for node v (n is the next node).

	Car	Train	Bus
Car	0	N.P.	N.P.
Train	PT(v) + TrW(v, n)	TrWTr(v, n)	TrW(v, n)
Bus	PT(v) + BW(v, n)	BW(v, n)	BWB(v, n)
Foot	PT(v)	0	0
Taxi	PT(v) + TaW(v)	TaW(v)	TaW(v)
Metro	PT(v) + MW	MW	MW

	Foot	Taxi	Metro
Car	N.P.	N.P.	N.P.
Train	TrW(v, n)	TrW(v, n)	TrW(v, n)
Bus	BW(v, n)	BW(v, n)	BW(v, n)
Foot	0	0	0
Taxi	TaW(v)	0	TaW(v)
Metro	MW	MW	0

Legend:
N.P. Not Possible, because a user cannot take the car at intermediate nodes;
PT is the Parking Time;
MW is the constant METRO_WAITING_TIME
TaW is the Taxi Waiting Time;
TrW & BW represent the waiting times for trains and buses as computed before;
TrWTr & BWB represent the waiting times for changes of homogenous means.

Table 2 Intra-node waiting times: in columns are the previous means, in rows the next. Fixed change times are omitted.

Traversing a node with the same means is not always a 0-cost operation. More precisely, this is true for means that involve different lines, thus trains and buses in our case study. For example, using the above notation, a user could reach station v from another station w with a bus line which does not continue to station n. In this case the user has to stop at station v and wait for a bus to station n. The problem is further complicated by the fact that there may be other bus lines connecting w to n through the node v and/or other different lines connecting v to n. As a consequence, it must be taken into account the probability the user takes each line of a bus/train connection. So, for trains and buses we have implemented two specific functions BWB(v, n) and TrWTr(v, n), as specified below:

$$BWB(v,n) = BW(v,n) * Pr\{v,n\} \tag{6}$$

[2] Eventually this feature can be disabled with an advanced interface option.

[3] It may be worth to rememeber that the frequency is not constant throughout the day, but may change according to real schedules.

[4] It is the eventual time it takes the user to go on foot from the arrival place to the departure place within the same node.

where *Pr{v,n}* is the probablity that the user arrives in node v with a bus line that does not continue to n. This last probability is obtainable from information dynamically stored in each node, that records the lines that the user coming from the previous node may take and the corresponding probabilities.
In all cases, we suppose that the system does not know buses, metro nor trains schedules (i.e. it knows only the frequencies of the services). This is realistic in many cases (e.g. the local bus company provides only frequencies, not precise schedules, or an user could arrive at a station without knowing the trains departure time), but may also be arbitrary (e.g. in a low-frequency line, users usually know the precise schedules of the trains or of the buses). Currently users can disable the intra-node change times option for the starting node or for all the nodes. However further better solutions could be studied to overcome this problem, probably by giving users some finer control over the computational process.

4. Results and conclusions

In order to validate our algorithm we run the program on different platforms and registered the performance in terms of time spent to get the final result. Table 2 shows the avarage results we got by running the server program within the jdk1.1 VM on a workstation Ultra Sparc1 140MHz with 64 Mbyte of memory and within the jdk1.2 VM on a PC Windows NT with 32 Mbyte of memory and a 200 MHz cpu. On the Sun platform we have also tried a native compiled version of the Java program, using Toba[10].

	Platform-Execution method		
MEANS	Ultra Sparc1-Interpreted code		
	1 zone	3 zones	5 zones
Bus	0,59	1.66	2.4
Bus & train	0.6	1.75	2.6
Car	0.45	1.25	1.71
Tax, car and foot	0.6	2.47	3.96
All	0.81	3.85	6.15
	Ultra Sparc1-Compiled code		
Bus	0.58	1.78	2.77
Bus & train	0.67	2.06	3.31
Car	0.38	1.09	1.69
Tax, car and foot	0.57	2.54	4.44
All	0.86	4.61	7.75
	NT-PC-Interpreted code		
Bus	0.58	1.78	2.77
Bus & train	0.67	2.06	3.31
Car	0.38	1.09	1.69
Tax, car and foot	0.57	2.54	4.44
All	0.86	4.61	7.75

Table 3 Execution time in seconds.

We developed the tests by varying the number of zones involved in the simulation and by varying the pool of selected means of transport, as shown in Table 3.
Further work may be spent to address the problem of how to reach many target points in the city in minimum time (travelling salesman problem).

References

[1] C.Mangione Just in Time for Java vs. C++, NC World January 1998 available at http://www.idg.net/idg_frames/english/content.cgi?vc=docid_9-34575.html
[2] R.N.Horspool, J.Corless Tailored Compression of Java Class Files, Software-Practice and Experience, Vol.28, No.12, 1998, pp. 1253-1268.
[3] B. Venners Inside the Java Virtual Machine, McGraw Hill, New York, NY, 1998.
[4] K. Arnold,J. Gosling The Java Programming language, Addison-Wesley, 1997
[5] T.Lindholm, F.Yellin The Java virtual Machine specification, Addison-Wesley, 1997
[6] N.Christofides *Graph Theory An Algorithmic Approach.*, Academic Press, 1975
[7] Linee Urbane AMT-FS Orari-Frequenze e Tariffe. Edizione 1998-1999
[8] E.W.Dijkstra A note on two problems in connection with graphs. *Numerishe Mathematik*, Vol.1, 1959, pp. 269-271
[9] Pearl, J. Heuristics: Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley, Reading, MA, 1984
[10] T.A. Proebsting, G. Townsend, P.Bridges, J.H. Hartman, T. Newsham and S.A. Watterson. Toba: Java for applications – a way ahead of time (WAT) compiler. In Proceedings of the 3rd Conference on Object-Oriented Technologies and Systems, 1997.