Public Transport Route Finding using a Hybrid Genetic Algorithm

Liviu COTFAS, Andreea DIOSTEANU
Academy of Economic Studies, Bucharest, Romania
liviu.cotfas@ase.ro, andreea.diosteanu@gmail.com

In this paper we present a public transport route finding solution based on a hybrid genetic algorithm. The algorithm uses two heuristics that take into consideration the number of transfers and the remaining distance to the destination station in order to improve the convergence speed. The interface of the system uses the latest web technologies to offer both portability and advanced functionality. The approach has been evaluated using the data for the Bucharest public transport network.
Keywords: *Route Finding, Evolutionary Algorithms, Hybrid Genetic Algorithm*

1 Introduction

Standard shortest path algorithms allow finding the shortest routes in networks featuring static and deterministic links. Dijkstra [1] and A* algorithms are widely used and considered the most efficient algorithms for the classic route finding problem. Several issues exist when applying such algorithms to public transport route finding for which links are time-dependent.

Several approaches based on modified Dijkstra algorithms like A* currently exist as well as algorithms that focus on speed, rather than on obtaining an exact solution. In [2] the authors present a genetic algorithm, called RGEAwK for identifying routes in the public transport network. To improve performance, the algorithm uses a transfer matrix and a minimal distance matrix. As in many other systems, the user is only allowed to search for routes between specified stations. In many cases the user might not know which stations close to the start and end points of his journey should be chosen in order to obtain an optimal route. Statically storing the transfer nodes is also used in [3]. A drawback for this approach is that the system has to compute large transfer areas as different users can choose different maximum walking distances.

In this paper we present a hybrid genetic algorithm that uses heuristics to improve the time required for generating a solution. The algorithm automatically determines the best route from the stations that are close to the user's location. Transfer areas are dynamically identified by taking into consideration the maximum walking distance specified by the user and the amount of walking already included in the route. We have chosen a heuristic approach because users need to find routes in near real-time. Moreover, using genetic algorithms offer the advantage of automatically generating several possible solutions.

The rest of paper is organized as follows: the second section includes a short presentation of evolutionary algorithms, while in the third section we introduce an efficient hybrid genetic algorithm. The fourth and fifth sections present mTripAssistent, a novel itinerary recommender system based on semantic web service composition. The system generates recommendations for Points of Interest – POI using a collaborative filtering algorithm that takes into account the user profile and the current context. The user can easily identify routes to the POIs using the algorithm described in this paper. The last section concludes the article and presents the future research guidelines.

2 Evolutionary Algorithms

Genetic Algorithms – GA [4] are a particular class of Evolutionary Algorithms – EA, that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover. Other related algorithms, also used in optimization problems are Simulated Annealing – SA, Ant Colony Optimization – ACO and Particle Swarm Optimization – PSO. Their application scope includes eco-

nomics, game theory, pattern recognition, neural networks and fuzzy theory. Genetic Algorithms have also been applied successfully for private car route finding like shown in [5].

The two most important aspects of any genetic algorithm implementation are the fitness evaluation function and the reproduction scheme that must be mutually compatible. Genetic algorithms try to eliminate the bad traits from the population using the process of selection. The good traits survive and are mixed by crossover to form potentially better individuals. The mutation operation ensures that diversity is not lost in the population, so the algorithm can continue to explore the solution space. Several approaches exist in respect to how the population P_t is built starting from the previous one, P_{t-1}. The Queen-Bee [6] approach relies on combing all the individuals in the population with the one that has the highest fitness value. The Elitist approach automatically preserves all the best chromosomes from the previous generation and discards the rest. All the new chromosomes are created using mutation and crossover from the selected individuals.

CHC introduced by [7] was proposed as a GA that combines the advantages of preserving the best individuals found so far between generations with a highly disruptive crossover in order to produce offsprings that are maximally different from their two parents. Most implementations add a new bias against combing individuals which are more similar. Instead of using the mutation operation for assuring diversity, a restart process is used each time the population converges.

Given the fact that many optimization problems, including the route finding one, can have multiple conflicting objectives, several approaches based on genetic algorithms have been proposed [8]. The easiest solution is to combine the individual objective functions using utility theory or the weighted sum method. In practice, it can be difficult to precisely and accurately select the weights. A more complex approach is to determine the entire Pareto-optimal set of solutions or a sub-set of it. The Pareto optimal set of solutions consists of all non-dominated solutions, for which any improvement of one of the objectives, means a decrease in at least another objective.

3 Route finding using a Hybrid Genetic Algorithm

Finding routes in public transport networks presents several particularities compared to normal route finding in graphs such as:

- public transport network contain **transfer areas**, representing areas where travelers can walk on foot between stations;
- transport modes have variable **time schedules** that must be taken into consideration in order to minimize waiting time;
- modes can be of different types including subway, bus, etc. and can have **different prices**;

Public transport networks can be seen as graphs in which edges can be traversed only at certain moments in time based on transport modes schedule. Due to the fact that the network is time-dependent, the problem of identifying routes with minimum travel time in the public transport network is NPc [9]. The network can be represented as a directed graph as follows:

$$G = \langle S_{PT}, L_{PT} \rangle \quad (1)$$

where:

$$S_{PT} = \{S_i | \, i \in \{1, 2, \dots N_{PT}\}\}$$

is the set of vertices representing the public transport stations. Each node is associated a natural number from *1* to N_{PT}, representing the number of the station.

$$L_{PT} = \{(S_i, S_j) | \, S_i, S_j \in S_{PT} \text{ and there is at least one mode from } S_i \text{ to } S_j\}$$

is the set of directed edges representing the links between stations.

We also define by M_{PT} the set of modes in the public transport network. Figure1 presents the public transport network in Bucha-

rest with over 2800 stations, as well as a transfer area.

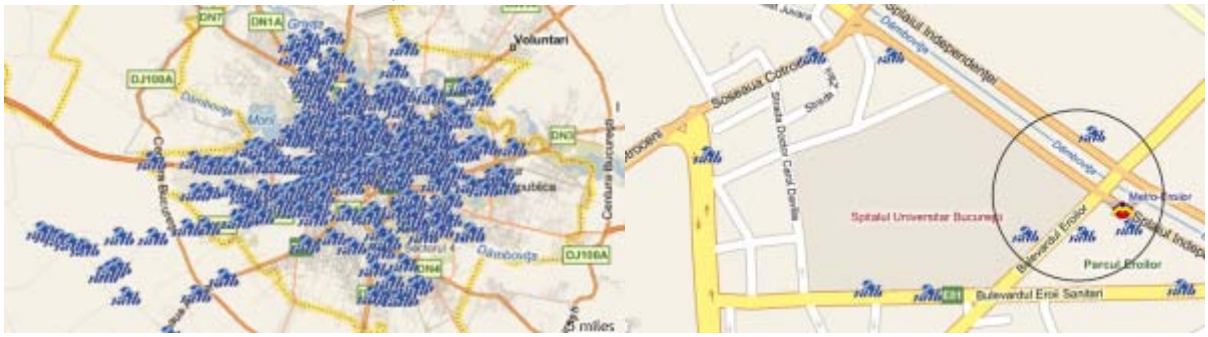

Fig. 1. a) The public transport network in Bucharest b) A transfer area

The inputs of the algorithm are the start time T_S, the start location as GPS coordinates L_S, the destination location as GPS coordinates L_D and the maximum walking distance $walk_{max}$. The first step is to identify the stations close to L_S that can be used as starting points for the journey.

In many cases choosing the station that is closest to the user position, leads to sub optimal results. Therefore in order to identify the possible start stations for the itinerary, we first find the closest station to the destination D' by calculating $\min(dist(S_i, L_D))$. By $dist(X, Y)$ we represent the walking distance between two locations. Using geographic functions, D' is searched only in the circle area defined by $walk_{max}$. We than compute the set $S = \{S_i | d(S_i, L_S) < walk_{max} - dist(D', L_D)\}$ of possible start stations.

Because in large networks the algorithm can require a significant number of generations to reach a good solution, we use two heuristics to enhance the convergence speed:

- **Minimum distance:** Moving to a station closer to destination offers higher chances of finding a good route. Therefore, when constructing routes, we will choose the station that will offer the maximum decrease in the Euclidian distance to the destination with a probability $prob_{MinDistance}$. The A* algorithm uses a similar approach to improve the speed as compared to the Dijkstra algorithm.
- **Minimum number of transfers:** In many cases, transfers increase both the walking distance and the total travel time T_t. Therefore each time we can make a transfer to other stations, around the current one, we will make transfer with a probability $prob_{MinTransfers}$.

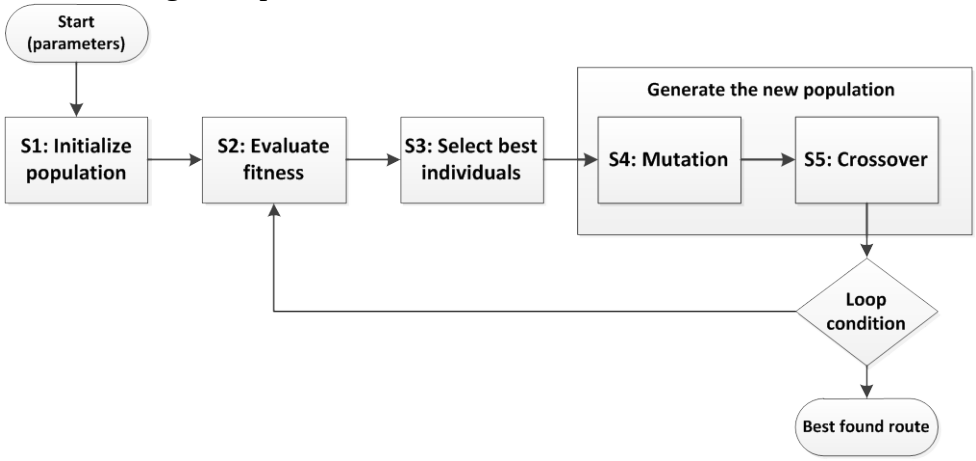

Fig. 2. Route finding using the Genetic Algorithm

The main steps of the algorithm are presented in Figure 2. The algorithm is run until a maximum number of generations, G_{MG} is reached or until the best found solution doesn't change for G_{CG} consecutive generations. Depending on the purpose G_{MG} can be adjusted either for accuracy or speed. In order to reduce the number of steps we always preserve the best chromosomes from one generation to the other. This approach is also known as Elitist Genetic Algorithm – EGA. Therefore we define by P_g and E_g the normal population and the population of non-dominated solutions in generation g, where $g < G_{MG}$. We present below the steps needed to generate the itinerary.

Initialization: For $g = 0$, we create the initial population P_0, with a number of N_R routes called chromosomes. We also initialize $E_0 = \emptyset$. In order to limit the number of required algorithm iterations, all the routes in the initial population are generated valid. From each possible start station we generate a number of $N_R/|S|$ routes using the above mentioned heuristics.

We consider that a route valid if it starts from a station that belongs to the set S, it does not include cycles and the total walking distance is smaller than $walk_{max}$. The individual routes are called chromosomes and the routes between consecutive stations are called genes. Therefore the optimality of a chromosome is characterized by the composing genes and their order.

Example: C_1 = (1, 2, 6, 9, 4); C_2 = (1, 8, 9, 4); C_3 = (1, 5, 6, 2, 3, 4);

Evaluation: We evaluate all N_R trips based on the following three criteria. For route j we compute NC_j the number of changes, T_j the travel time and $walk_j$ the total walking distance. Transfers are not separately taken into consideration in the evaluation step because they are evaluated based on the components, a line change and walk distance, that both affect the total time of the route.

Selection: We have chosen an elitist approach in which we copy all non-dominated solutions from P_g in E_g. We remove all other solutions from P_g and we automatically include all solutions from E_g in P_{g+1}. Using mutation and crossover we generate $N_R - |E_g|$ solutions that are added to P_g in order to have $|P_{g+1}| = N_R$.

Mutation: Adds random variation to the evolving population. We randomly choose 2 stops in the route and we remove all the links between them. We than complete the itinerary by generating a new route between the two stops.

Example: C_2 = (1, 5, 6, 2, 3, 4) → C^*_2 = (1, 5, 7, 4)

Crossover: Combines the features of two parent chromosomes to form new children by swapping corresponding route segments of the parents. We first randomly select two stations that are common in both parent routes. We than swap the route segments between the stops. Afterwards, we run a repair procedure to eliminate any cycles that might have appeared in order to maintain the routes valid.

Example: *C_1 = (1, 2, **6**, 9, 4)*, *C_3 = (1, 5, **6**, 2, 3, 4)* → *C^*_1 = (1, 2, **6**, 2, 3, 4)*, *C^*_3 = (1, 5, **6**, 9, 4)*

The results have been obtained on the public transport network in Bucharest with over 2800 stations. 82 lines, representing almost the entire network of public transport network, were taken into consideration for the analysis. Figure 3 compares the evolution of the best fitness value over 40 generations when using the Classical GA approach versus the Hybrid one that uses the minim distance and minimum number of transfer heuristics. It can be seen that the hybrid approach in which $prob_{MinDistance} = 0.7$ and $prob_{MinTransfers} = 0.2$ have been used converges faster and offers better results in a smaller amount of time.

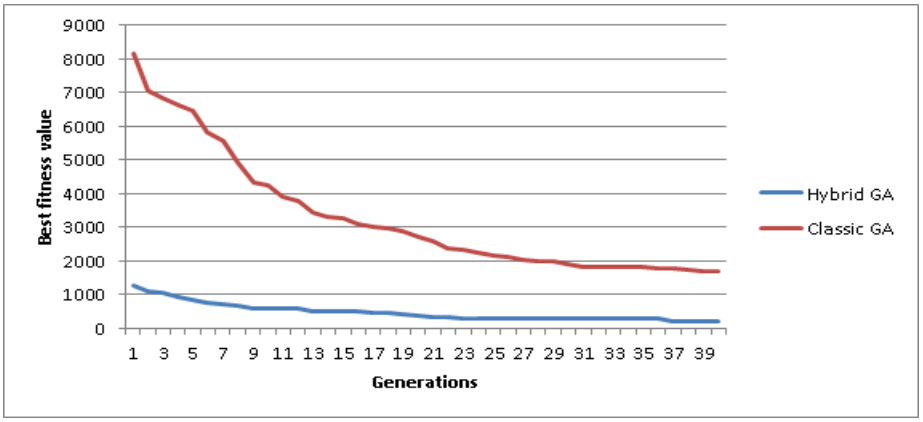

Fig. 3. Comparison between the convergence speed for Classic and Hybrid GA

In order to evaluate the performance, random sets of stations have been selected and for each set both algorithms have been run 10 times. The combined results are presented in the figure above.

4 Design and Implementation Architecture

The proposed solution relies on a multi-tier paradigm for both the server and the client implementations as shown in Figure 4. Even though a client only architecture would offer several benefits, such as the possibility to work entirely offline, it is not a feasible option for our approach due to the big amounts of data that are used by the recommendation algorithm. Therefore, we have chosen a mixed approach in which the resource intensive computations are performed on the server, while the simple ones are performed directly on the client. Client implementations rely on local persistence of data in order to avoid unnecessary server requests.

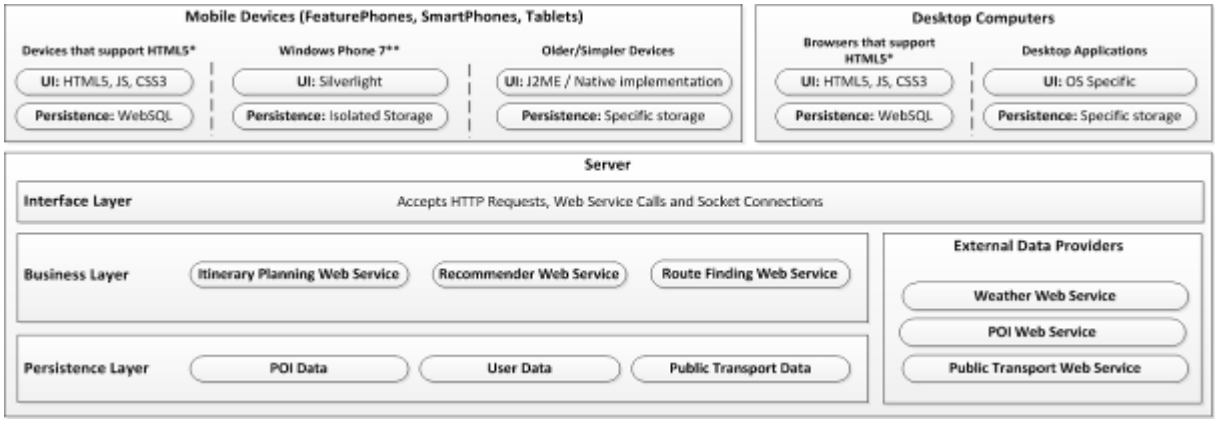

Fig. 4. mTripAssistent multi-layer arhitecture

Section 5 presents a client implementation based on HTML5, using WebSQL as a client persistence approach.

5 Mobile Web User Interface

As shown in the previous section, the proposed architecture implements an Interface Layer, supporting HTTP requests, Web Service Calls and Socket Connections. Thanks to the multiple communication methods supported by this layer, client side applications can be implemented using a wide array of technologies.

A common problem for mobile application developers is the increasing fragmentation of the mobile device market in which a large number of manufacturers use several different operating systems. As each operating system requires a different development approach, the cost and time needed to build applications increases. Even though, Java Micro Edition [10] was considered for a long

time the solution for creating platform independent applications for mobile devices, it currently doesn't support the iPhone and Android mobile operating systems. Using standard web technologies to create applications is currently emerging as a solution to develop applications that run on all mobile platforms. The reference client implementation for mTripAssistent shown in Figure 5 relies on the latest web technologies such as WebSQL [11], WebStorage [12] and Offline Application Cache to offer portability and a rich user experience. All three technologies are important parts of the new HTML5 standard [13]. In the future, the local persistence can be changed from WebSQL to Indexed Database API [14]. In order to determine and monitor the position of the user, the W3C GeoLocation API [15] has been used.

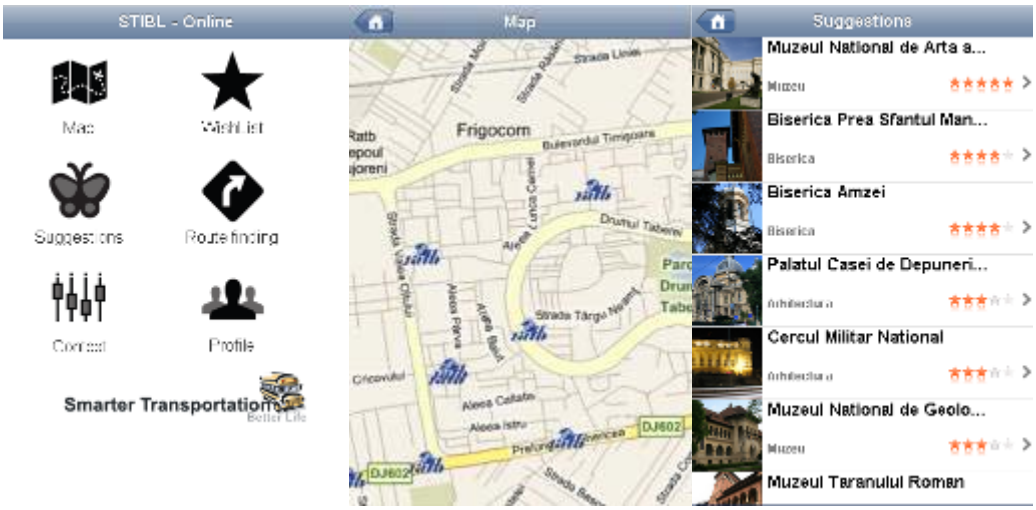

Fig. 5. mTripAssistent interface developed using HTML 5

Because the schedule provided by the public transport companies can differ from the real one due to several reasons, a crowd sourcing approach is under implementation in order to determine the actual travel time between stations at different hours and week days. The information is automatically collected by the client application and transferred to the server component of the system for aggregation.

6 Concluding remarks

In this paper we have presented a novel public transport route finding solution. From the architectural point of view, the proposed system is design to support a wide array of technologies for building client side applications. The proposed client side reference implementation, built using the latest web technologies such as WebSQL and Offline Application Cache, delivers a uniform experience on all the devices that support the new HTML5 standard. In the future we want to compare the results for the current algorithm with the results of a CHC GA implementation. Also, more options can be include in the algorithm, like finding routes that are close to interesting Points of Interest – POI using geographic functions to evaluate the distance.

Acknowledgement

This article is a result of the project „Doctoral Program and PhD Students in the education research and innovation triangle". This project is co funded by European Social Fund through The Sectorial Operational Programme for Human Resources Development 2007-2013, coordinated by The Bucharest Academy of Economic Studies.

References

[1] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, 1959, p. 269–271.

[2] A. Piwonska and J. Koszelew, "Evolutionary Algorithms Find Routes in Public Transport Network with Optimal Time of Realization," *Transport Systems Telematics*, 2011, p. 194–201.

[3] C. Jun, "Route Selection in Public Transport Network Using GA," in *Proc. Esri User Conference*, http://gis.esri.com /library/userconf/ proc05/papers/pap1874.pdf.
[4] C.H. Lin, J.L. Yu, J.C. Liu, C.J. Lee, "Genetic Algorithm for Shortest Driving Time in Intelligent Transportation Systems," in *Proc. of the 2008 International Conference on Multimedia and Ubiquitous Engineering*, pp. 402-406, 2008.
[5] C. Lin, J. Yu, J. Liu, and C. Lee, "Genetic Algorithm for Shortest Driving Time in Intelligent Transportation Systems," 2008, pp. 402-406.
[6] S.H. Jung, "Queen-bee evolution for genetic algorithms," *Electronics Letters*, vol. 39, 2003, p. 575–576.
[7] L.J. Eshelman and J.D. Schaffer, "*Preventing Premature Convergence in Genetic Algorithms by Preventing Incest*," *ICGA*, R.K. Belew and L.B. Booker, eds., Morgan Kaufmann, 1991, pp. 115-122.
[8] A. Konak, D. Coit, and A. Smith, "Multi-objective optimization using genetic algorithms: A tutorial," *Reliability Engineering & System Safety*, vol. 91, Sep. 2006, pp. 992-1007.
[9] H.M. Safer, J.B. Orlin, "Fast approximation schemas for multicriterial combinatorial optimization", *Tehnical Report No. 3756-95*. Cambridge, Massachusettes Institute of Tehnology, Sloan School of Management (1995).
[10] Oracle, "Java Me," 2011. [Online]. Available: http://www.oracle.com/ technetwork/java/javame/overview/. [Accessed: January. 15, 2010].
[11] W3C, "Web SQL Database," 2011. [Online]. Available: http://www.w3.org/ TR/webdatabase/.[Accessed: January. 15, 2010].
[12] W3C, "Web Storage," 2011. [Online]. Available: http://www.w3.org/TR/ webstorage/. [Accessed: January. 15, 2010].
[13] W3C, "HTML5," 2011. [Online]. Available: http://dev.w3.org/html5/spec/ Overview.html. [Accessed: January. 15, 2010].
[14] W3C, "Indexed Database API," 2011. [Online]. Available: http://www.w3.org/TR/IndexedDB/. [Accessed: January. 15, 2010].
[15] W3C, "Geolocation API," 2011. [Online]. Available: http://www.w3.org/TR/ geolocation-API/. [Accessed: January. 15, 2010].

Liviu Adrian COTFAS is a Ph.D. student and a graduate of the Faculty of Cybernetics, Statistics and Economic Informatics. He is currently conducting research in Economic Informatics at Bucharest Academy of Economic Studies and he is also a Pre-Assistant Lecturer within the Department of Economic Informatics. Amongst his fields of interest are location based services, recommender systems, geographic information systems, evolutionary algorithms and web technologies.

Andreea DIOŞTEANU has graduated the Faculty of Economic Cybernetics, Statistics and Informatics in 2008 as promotion leader, with an average of 10. She is currently conducting research in Economic Informatics at Bucharest Academy of Economic Studies and she is also a pre-Assistant within the Department of Economic Informatics and .NET programmer at TotalSoft. During the bachelor years she participated in many student competitions both at national and international level obtaining a lot of first and second prizes. The most important competitions she was finalist in were Microsoft International Imagine Cup Competition, Software Design section (national finalist); Berkley University and IBM sponsored ICUBE competition were she qualified for the South Eastern Phase-Novatech. Furthermore, she also obtained the "N.N Constantinescu" excellence scholarship in 2007-2008 for the entire student research activity.