Integrating System Optimum and User Equilibrium in Traffic Assignment via Evolutionary Search and Multiagent Reinforcement Learning

Ana L. C. Bazzan[1] and Camelia Chira[2]

[1] Programa de Pós-Graduação em Computação
Universidade Federal do Rio Grande do Sul, Brazil
bazzan@inf.ufrgs.br
[2] Technical University of Cluj-Napoca
Baritiu 26-28, Cluj-Napoca, Romania
camelia.chira@cs.utcluj.ro

Abstract. Traffic assignment is fundamentally a tool for transportation planning. It allocates trips within the traffic network. However, modern uses of traffic assignment also include shorter time horizons and even real-time use (e.g., for route recommendation). In the latter case, it is interesting to recommend routes that are as close as possible to the system optimum. To compute an approximation of the optimal traffic assignment, we use a hybrid approach, in which an optimization process based on an evolutionary algorithm is combined with multiagent reinforcement learning. This has two advantages: first, the convergence is accelerated; second, the multiagent reinforcement learning resembles the adaptive route choice that drivers perform in order to seek the user equilibrium. Thus, the hybrid approach aims at incorporating both the system and the user perspectives in the traffic assignment problem. Results are encouraging: the combination of the evolutionary approach and the multiagent reinforcement learning accelerates the computation and delivers an efficient assignment in terms of travel time.

1 Introduction

Traffic assignment is fundamentally a tool for transportation planning, whose aim is to do the assignment of traffic onto the routes of a network. Besides planning, modern uses of traffic assignment also include shorter time horizons and even real-time use, in order to give route recommendation to drivers. This task is not trivial though. First, this process has to consider the fact that one trip has influence on others. Second, if a driver receives a route guidance that significantly deviates from a reasonable travel time, this driver is likely to reject future recommendations.

There are several ways to deal with the traffic assignment problem (TAP). One of these is based on the fact that drivers perform experimentation in order to

assess the utility of a given number of alternative routes: drivers adapt their route choices for the n-th day based on the travel time (or any other utility function) of the previous days. This is the basis of the the so-called user equilibrium, in which, as stated by Wardrop [1]: "under equilibrium conditions traffic arranges itself in congested networks such that all used routes have equal and minimum costs while all those routes that were not used have greater or equal costs". This is Wardrop's first principle, also known as Wardrop's equilibrium or user/Nash equilibrium. Wardrop's second principle states that the average trip time is a minimum. The assignment resulting from this principle can be thought of as one that describes the network at its best operation, in which congestion is minimized when drivers are told which routes to use. Obviously, this is primarily useful to transport planners in trying to manage the traffic in order to achieve an optimum social equilibrium. However, it also has applications in the operations of a network: while in a real, congested, urban traffic system the flows observed are more likely to be closer to an user optimum (though this varies greatly), traffic authorities strive to obtain the system optimum flow because the latter is more efficient. Indeed, there are studies that precisely quantify the cost of the anarchy, i.e., the situation in which each driver seeks to minimize its travel time independently, as discussed in Section 2.

One way to achieve the system optimum is to impose routes (this is possible for instance in restricted systems, where a centralized controller makes decision as in logistic centers or railway systems). Another is to impose tolls, congestion pricing, rotation systems, delivery hours and other measures that aim at persuading drivers to change routes. However, these tend to be less popular. A third one is now possible, with the increasing availability of traffic information (e.g., Waze). However, because not everyone has access to such information or because some are new in town, part of the drivers still perform experimentation.

In this paper we take advantage of the fact that more and more information is being collected by some actor in the traffic system (be it a traffic authority or an independent actor such as Waze or Google traffic), and that route guidance is becoming a reality. In order to recommend routes, an assignment must be computed. It is then in the interest of the collectivity that such guidance is made in a way to spread the traffic flow over the network. We propose a hybrid traffic assignment method, in which the system optimum is computed via a genetic algorithm (GA). To account for drivers experimentation, some solutions of the pool of solutions are composed by routes that would have been computed by drivers themselves in their process of seeking the user optimum. In our case this is computed via reinforcement learning. Because each choice is likely to affect the reward of many others, this is a typical multiagent reinforcement learning problem. Some variants of this traffic assignment method are proposed, in which we compute an approximation of the system optimum, approximations for the user equilibrium, and a hybrid of these two, called here GA+QL.

The next section introduces background concepts and methods on the TAP, and also discusses related works. Section 3 describes the proposed approach and

the scenario used to illustrate it. Results are shown and analyzed in Section 4, while Section 5 presents the concluding remarks.

2 Background and Related Work

2.1 Network representation and edge costs

A road network can be represented as a graph $G = (V, E)$, where V is the set of vertices that represent the intersections of the network, and E is a set of directed edges, describing the existing road segments as directed connections between pairs of vertices. Each edge $e \in E$ has a cost c_e, which is given by a function that considers attributes such as length, toll, free-flow speed (and hence, free-flow travel time), capacity, flow, etc. A route or path p is defined by a sequence of connected vertices $(v_0, v_1, v_2, ...)$. The cost of each p is the sum of the costs of all edges e that connect these vertices.

One of the inputs of the TAP is a so-called origin-destination (OD) matrix, which contains the demand (number of trips) for each OD pair or flow.

Given such OD matrix, an assignment is generally done in a macroscopic way, i.e., vehicles are considered in an abstract way. One well known abstraction is the concept of volume-delay functions or formulas (VDFs), also known as edge or link performance formulas or cost-flow relationship. These functions abstract physical measures such as speed, gap between vehicles etc. in a relationship that accounts for congestion effects, i.e., how over-capacity in the edge affects the speed and travel times (costs of delays).

In short, a VDF takes as input the flow in the edge (no matter where this flow comes from and goes to), and outputs the corresponding edge travel time, accounting for delays that may occur. As an example of a VDF, one can consider the following: $t_e = t_{e_0} + 0.02 \cdot q_e$. Here, t_e is the travel time on edge e, t_{e_0} is the travel time per unit of time under free flow conditions, and q_e is the flow using edge e. This means that the travel time in each edge increases by 0.02 of a minute for each vehicle/hour of flow. Other more sophisticated VDFs exist (see Chapter 10 in [2] for instance).

2.2 User equilibrium and system optimum

As mentioned in the previous section, the purpose of traffic assignment is manifold, and there are many ways to perform this task. Next, we discuss the concept of Nash or user equilibrium (UE), and that of the system optimum (SO).

The UE is achieved by assuming that each user performs adaptive route choices until all used routes between an OD pair have equal and minimum costs while all those routes that were not used have greater or equal costs [1]. It must be said that, while this process aims at reproducing drivers behaviors, in the practice of traffic engineering this is performed by a central authority as a way to "simulate" the day-to-day choices of a number of drivers in various flows until an equilibrium is found.

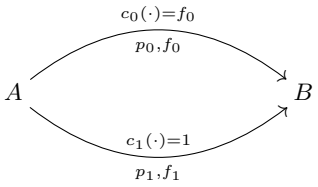

Fig. 1. Pigou's example.

The assignment that leads to the SO is computed by an optimization procedure, which can be done by minimizing the overall travel time on the edges.

To illustrate the difference between the UE and the SO we show a simple example due to Pigou (Figure 1). This considers a non-atomic flow $f = 1$, representing a large population of network users, each choosing independently between the two paths p_0 and p_1 that take from A to B. Flow on p_0 is f_0 and flow on p_1 is f_1 with $f_0 + f_1 = 1$. Each edge e has a VDF $c_e(\cdot)$ which computes the delay or cost in the edge. This cost for a given e is $c_e = c_e(\cdot) \cdot f_e$. The total cost is $C = \sum_{e \in E} c_e$.

In the example in Figure 1, p_1 represents a route with fixed cost of 1, while p_0 has a cost that depends on its flow f_0. For Pigou's example, the UE is that every user of the flow selects path p_0 because it is never worse than p_1, even if the whole flow uses p_0. This equilibrium has a total cost $C = f_0 \cdot f_0 + f_1 \cdot 1 = 1 \cdot 1 + 0 \cdot 1 = 1$.

However, the social or system optimum is just $3/4$, which is computed by minimizing $C = f_0 \cdot f_0 + f_1 \cdot 1$, i.e., $f_0 \cdot f_0 + (1 - f_0) \cdot 1$. The solution is $f_0 = f_1 = 1/2$ which gives $C = 1/4 + 1/2 = 3/4$. This shows that there is a degradation in performance caused by selfish routing, which can be measured by the so-called price of anarchy [3, 4].

One important point regarding the UE and SO refers to the complexity of their computation. For the former, this involves finding the Nash equilibrium. This means that it is not possible to find equilibrium flows algebraically, except for very simple cases (e.g., a few edges connecting few OD pairs). Thus, approximate solutions to the UE were proposed. See [5] for a general discussion and Chapter 2 in [6]. Methods to approximate the UE follow a general pattern: (i) identify a set of routes that are attractive to drivers (e.g., using any shortest path algorithm); (ii) assign suitable proportions of the trip matrix to these routes; this results in flows on the edges of the network; (iii) search for convergence.

Regarding the computation of the SO, given that analytical methods to compute the exact solution (e.g., based on convex optimization) are not always feasible or efficient, metaheuristics can be used.

2.3 Other traffic assignment methods

There are other schemes to assign trips to the edges of a road network. For more details please refer to [2]. One of these schemes assumes no congestion, assigning

all trips to the route with minimum cost, on the basis that these are the routes travelers would rationally select. This procedure is known as "all-or-nothing" assignment. Although unrealistic, it is used in iterative methods, which aim at computing approximations of the UE.

One of these iterative methods loads the network incrementally in n stages, e.g., assigning a given fraction p_n (e.g., 10%, 20%, etc.) of the total demand (for each OD pair) at each stage. Further fractions are then assigned based on the newly computed edge costs. This means that new shortest paths are computed at each stage (since costs change). This procedure continues until 100% of the demand is assigned. It must be remarked that there is no guarantee that this algorithm converges to the UE, no matter how small each p_n is.

2.4 Related work

Due to lack of space we focus on two classes of works that are related: those that deal with combinations of metaheuristics and other kinds of techniques (not necessarily for the assignment problem), and those that address the problem of traffic assignment.

In [7], Buriol et al. proposed an approach based on a biased random-key genetic algorithm that finds a traffic flow that is both UE and the SO. However, they solve this by finding a set of edges in which a number of toll booths should be placed, i.e., they solve a different problem than we do, namely imposing additional costs to make the UE and the SO coincide.

Regarding the combination of metaheuristics and reinforcement learning, there are two works that are relevant here, even if none deals with the traffic assignment problem. Wolpert et al. [8] proposed a hybrid approach using reinforcement learning and simulated annealing, applied to the domain of minimizing the loss of communication of satellites data. Their approach works by interleaving the Boltzmann-distribution-based exploitation step of conventional simulated annealing with an exploration step, where each agent takes a step that its reinforcement learning algorithm recommends. In their case, the utility of the agent is set to the system utility, rather than a local perceived utility, as in the present paper. Their approach, if applied to the traffic assignment problem would mean that each driver would receive the average travel time as reward, which is unrealistic given that in reality an agent perceives its own travel time. Lima Jr. et al. [9] use Q-learning as exploration/exploitation strategy for GRASP (Greedy Randomized Adaptive Search Procedure) and GA. GRASP uses Q-learning instead of a greedy algorithm to improve the quality of the initial solutions that are used in the local search phase of the GRASP. In the GA, Q-learning was used to generate an initial population of high fitness and, after a determined number of generations, it was also applied to supply one of the parents to be used by the crossover operator. They have applied this approach to solve the traveling salesman problem. Thus the nature of the problem being solved is completely different. In particular, this is not a multiagent problem, in which there is competition for resources, as in the TAP.

The works discussed next all address the TAP.

Henn [10] proposes a fuzzy-based method to take the imprecision and the uncertainties of the road users into account. These predict costs for each path based on a fuzzy subset that can represent imprecision on network knowledge, as well as uncertainty on traffic conditions.

A GA combined with a microscopic simulation is discussed in [11]. In a microscopic approach, the actual (physical) movement of cars is simulated. This outputs the travel time per link and, as such, replaces the VDFs. This can be seen as an advantage. However, microscopic simulations take a long time to run, which is not feasible for the purpose of quickly computing a route to recommend. Thus, the performance in terms of simulation times is poor compared to the macroscopic approach used in the present paper.

Regarding the computation of the UE, a natural way to represent the problem of individual route choice is to model it using an agent-based modeling and simulation approach, such as in MATSim [12]. However, in this work no comparison is made to methods that approximate the UE thus, it is not possible to fully assess the efficiency of those results.

Route choice under various levels of information is turning a hot research topic due to the increasing use of navigation devices. The main question here is what happens if a certain share of drivers is informed and adapts. Examples for such research line can be found in [13] for a two-route scenario, or in [14] where a neural net-based agent model for route choice is presented regarding a three route scenario. One problem with these approaches is that their application in networks with more than a couple of routes between a few locations is not trivial. One first issue is that a set of reasonable route alternatives has to be generated. A further problem is that these consider one route as one complete option to choose. To address this issue, on-the-fly re-routing in a scenario with multiple origins and destinations was studied in [15]. Similarly to MATSim, here nothing is stated about the efficiency of the computation of the UE.

3 Methods

The basic idea of the approach proposed in this paper is that, if a central authority (as, e.g., traffic authority or even Google traffic) could compute routes and recommend it to drivers (based on the highest possible amount of traffic information that this authority may be able to gather), then such recommendation could guide the system towards its best operation. Thus, the approach requires that the SO is computed. Given that analytical methods to compute the exact solution to the SO (e.g., based on convex optimization) may not be possible, an approximation is computed by using an evolutionary algorithm. Here, a GA is employed. A solution for the TAP is the allocation of given portions of the flows to determined edges of a network. This means that we have to compute a path for each trip, to go from its origin to its destination. We remark that paths need not to be computed for each trip or driver individually, but, rather, for classes of trips as, e.g., those sharing the same OD pair.

Algorithm 1 Pseudo-code for the GA+QL approach

```
INPUT: population size, elite size, mutation and crossover probabilities (for the GA); learning rate, initial value and decay rate on ϵ (Q-learning)
INPUT: k shortest paths that are available as action selection for each OD pair
generate GA population
while generation < max_nb_generations do
    evolve (elistism, reproduction, mutation, crossover)
    // Q-learning:
    for each trip or group of trips do
        ε-greedy action selection: action is randomly chosen with probably 1 − ε or route is the one with highest Q-value
    end for
    for each trip or group of trips do
        simulate trip, collect travel time, update Q-table (Eq. 1)
    end for
    substitute GA's worst individual by one formed by the individual route choices resulting from QL action selection
end while
```

One important point that has motivated our approach is that giving recommendations simply from the point of view of the performance of the system may hit some drivers hard, since their individual travel times may increase. These will tend to unilaterally divert to other routes. Therefore, our approach takes this into account. When assembling the population of solutions that takes part in the selection process of the GA, our approach replaces the worst solution in the GA pool by one that is computed the way each individual driver would, if it would learn to select routes individually. This is done by means of reinforcement learning, with the central authority simulating the sequential choices of actions by the drivers. Henceforth we refer to this approach by GA+QL.

We remark that the underlying assumptions of our approach are the same used by any method that computes the UE or the SO, namely that a central authority knows the OD distribution of the flows, which is reasonable, given OD surveys, or online data, or a combination of both.

Regarding the RL, here Q-learning is used. In Q-learning, the update rule for each experience tuple $\langle s, a, s', r\rangle$ is given in Equation 1, where α is the learning rate and γ is the discount for future rewards.

$$Q(s,a) \leftarrow Q(s,a) + \alpha\ (r + \gamma\, max_{a'}\, Q(s',a') - Q(s,a)) \tag{1}$$

The goal is to learn a policy by interacting with an environment that gives a feedback signal to each driver. In our formulation, the reward is the individual travel time, i.e., the travel time of an individual trip, not the average travel time. The available actions are the selection of one among k routes that are computed using a k shortest paths (e.g., the algorithm proposed by Yen [16]) to travel from the trip's origin to its destination. Due to this formulation of the problem (which resembles a repeated game where there is just one state as in [17] and

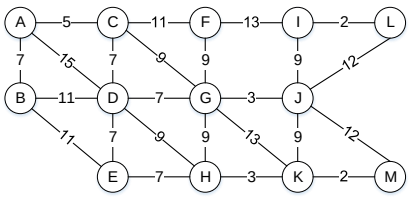

Fig. 2. Road Network

many others after them), it is necessary to represent just one state. Therefore, Equation 1 can be simplified, without the need of using a discount rate. While this simplifies the problem, considering a high number of agents in multiagent RL turns the problem inherently more complex. We defer this discussion to later in this section, where we show the case in which all agents follow their shortest paths (see discussion about Table 1).

For action selection we use the ϵ-greedy method: with probability ϵ, an action is selected, which has the highest Q-value, while, with probability $1-\epsilon$, an action is selected randomly. We follow the strategy of initializing ϵ with a high value, and multiplying it by a decay rate d. This allows high exploration at the beginning.

The part that refers to the computation of the SO by means of an evolutionary approach uses a GA with elitism. Each individual of the population of solutions is a set of shortest paths, one for each trip. Thus the length of the chromosome (that represents each solution) is the number of trips, and each position can take an integer value between 0 and $k-1$. A chromosome is, for instance, 3 7 0 6 6 4 ...0 1 (here for $k = 8$). Given a population containing p chromosomes, the GA evolves the population so that the average travel time (the fitness function) is minimized. The evolution is made in a standard way with selection for crossover pairing with probability c and a mutation probability p_m. The pseudo-code for the GA+QL is as Algorithm 1.

This optimization process is far from trivial because edges have travel times that depend on the number of trips using them, thus potentially each choice affects the travel times of many other trips.

A remark on scalability must be made here. At first glance, it seems that this method does not scale well with the number of trips, especially if k is large. Two, complementary, solutions for this are: (i) keep k low; in fact, k is typically not more than a couple of paths (either because there are indeed not many significantly different paths in real-world networks, or because drivers do not know them and/or their costs with accuracy); (ii) the number of trips to be put in the chromosome can be reduced by grouping them (e.g., a number of trips that have origins and destinations in the same districts of the network, and/or similar driving patterns can be grouped so that a path will be assigned for the group). These two measures greatly limit the size of the chromosome.

Table 1. $k = 4$ Shortest Paths and Their Free-Flow Travel Times (FFTT) for the Four OD Pairs.

OD	Trips	sh. path 1		sh. path 2		sh. path 3		sh. path 4	
AL	600	ACGJIL	28	ACGJL	29	ACFIL	31	ACDGJL	34
AM	400	ACDHKM	26	ACGJKM	28	ACGJM	29	ADHKM	29
BL	300	BDGJIL	32	BDGJL	33	BACGJIL	35	BACGJL	36
BM	400	BEHKM	23	BDHKM	25	BDEHKM	30	BDGKM	33

In order to illustrate the approach, a non-trivial traffic network is used, namely the one suggested in Chapter 10 of [2] (Exercise 10.1) and depicted in Figure 2. Henceforth this network is referred as OW network. In this network all edges are two-way. The network represents two residential areas (nodes A and B) and two major shopping areas (nodes L and M). The numbers in the edges are their travel times under free flow (the travel time for traversing the edge *under no congestion*). For the aforementioned scenario, authors in [2] use 1700 trips, i.e., this is the estimated demand from origins A and B to destinations L and M as depicted in the second column of Table 1.

If there were just a single trip using the network (hence, no congestion), the time taken for such a trip would be its free flow travel time. Table 1 shows these times if a trip is done using the shortest path ($k = 1$), the second shortest path ($k = 2$), etc. However, the assumption of no congestion is unrealistic in this scenario because the flow of 1700 trips cannot use the edges simultaneously without causing congestion. For this case, a VDF was given, which relates cost $c(q_e)$ at edge e to its flow q_e. Specifically, it is proposed that the travel time in each edge is increased by 0.02 for each trip/hour of flow ($t_e = t_{e_0} + 0.02 \cdot q_e$, as discussed in Section 2). That is why we use this simple VDF, and remind the reader that more sophisticated ones (e.g., exponential relationship between flow and delay) could be used as well. Using non-linear cost functions turns the problems of computing the UE and the SO algebraically harder; this reinforces our idea of computing approximations for these two quantities.

To give an idea of travel times in this case, if each trip is assigned its shortest paths, the average travel time for all trips would increase. In particular, trips in the OD pair AL are hit hard: instead of the expected travel time of 28, the average travel time for AL trips would be 114 (see Table 2). This explains why a good assignment is so important in this scenario.

Differently from two-route (binary) choice scenarios that are common in the literature, this one captures properties of real-world scenarios, like interdependence of routes with shared edges, heterogeneous demand, and more than a single OD pair. Hence, it is hardly possible to compute the UE algebraically.

4 Experiments and Results

To assess the efficiency of our method, the main performance measure is the same as used in [2]: travel times averaged over all trips and also over trips in

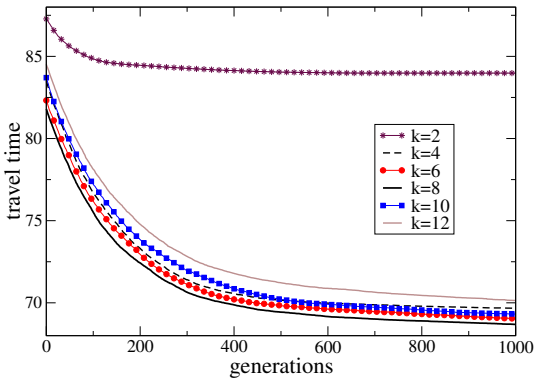

Fig. 3. Performance along generations for various values of k ($p_m = 0.001$ and $c = 0.2$).

each OD pair. We remark that results reported here for the GA, Q-learning, and their combination are averaged over 30 repetitions (for each condition). The standard deviation is very low (close to zero once the convergence was established; less than 5% for the Q-learning in the high exploration phase, i.e., only in the beginning), thus they are omitted from plots.

Regarding the parameters of the GA, a population of size 100 was used, with elitism (the 5 best solutions were transferred to the next generation without change). For the remaining 95 individuals, selection was done using crossover rate $c = 0.2$ and mutation rate $p_m = 0.001$. Note that these values were selected after extensive tests with other values. After these tests, it is possible to state that the crossover rate has low influence and that too high or too low mutation rates are deleterious. It seems that if mutation is higher than the one we are using, choices are too random and thus convergence is harder.

Still regarding parameters' values, Figure 3 shows a comparison of performance – in terms of average travel time – for various values of k, as this affects the performance of our method. If k is low, then there are few options of routes. Not only this: from Table 1 one can see that fewer edges are used in the shortest paths when $k = 2$ (as compared, e.g., with $k = 4$). The consequence is clear: less diversity in options for combinations of routes leads to many trips using the same edges. This leads to higher travel times for everyone. However, also high values for k are inefficient (too many choices; paths do not differ much). From Figure 3 one can conclude that $k = 8$ is a good compromise since higher values for k do not bring better performance and represent more computational effort.

Henceforth, all plots refer to the following values: $k = 8$, $p_m = 0.001$ and $c = 0.2$, and we now discuss the results in basically three situations. First, when only GA is used to compute an approximation for the SO. Second, when only Q-learning is used (this approximates the UE). Third, when the GA+QL approach is used. In the latter case, the worst solution in the pool of GA solutions is

replaced by one computed via Q-learning. Because a learning step (for each trip) in the Q-learning occurs in parallel with the GA (see Algorithm 1), plots refer to generations in the x-axis, even this is not fully correct in case the GA is turned off and only Q-learning is active.

The average travel time over the 1700 trips is shown in Figure 4, in which all situations are plotted together just to give a full panorama. Figure 5 shows the same plots, grouped by value of the learning rate α. In these figures, when only GA is used, it is depicted as the full black line for comparison. Lines in other colors refer to using only Q-learning (due to the number of curves, only some of these are shown), while the cases in which GA+QL is employed are plotted using symbols. The various curves refer to various values of the learning rate α and the decay d (the multiplicative quantity over ε) that is used for action selection. In Figure 4 one sees that the convergence (i.e., to average travel time around 70) is achieved much later (after generation 600) than in the other cases. This can be seen in Figure 5. As expected, different values of α and decay d result in different convergence times.

In general, looking at the plots that refer to the use of Q-learning only in the three sub-figures in Figure 5, one concludes that low values of learning rate α (i.e., the weight on the current reward, see Eq. 1) do not lead to good performance due to the high weight on the immediate reward (rather on the Q-value).

When $\alpha = 0.1$ (Figure 5, upper), the plots that refer to the use of only Q-learning lead to bad performance (higher average travel times) because nothing is learned after exploration is over. Learners get stuck at a bad choice of routes and the average travel time no longer changes (see, e.g., the curve for $\alpha = 0.1$ and $d = 0.995$ when only Q-learning is used). With these values, since the Q-learning alone shows poor performance, its combination with the GA is not able to improve the travel time significantly.

When $\alpha = 0.9$ (Figure 5, lower), more weight is put on the Q-value (rather than on the immediate reward). The plots that refer to the use of only Q-learning show that in these cases the average travel time drops. The time taken for this to happen of course depends on the length of the exploration phase. For higher values of d, action selection is random for a longer time and this is seem in the plots (e.g., see the curve for $\alpha = 0.9$ and $d = 0.995$ in Figure 5).

The combination of Q-learning with the GA depicts a very good performance in these cases. The Q-learning at individual level needs not to explore much to provide the GA good alternative routes. For instance, if $\alpha = 0.9$ and $d = 0.9$ as in the lower sub-figure of Figure 5 (this is combination of parameters that results in the lowest travel times), the average travel time of 67.2 is achieved already around generation 50. Travel times below 67 are achieved around generation 600. And the travel time of 72 (this is the result provided by the incremental method) is reached at generation 12. Values of $\alpha = 0.5$ have balanced performances (see Figure 5, middle).

In short, one can say that if the GA is used alone, the convergence happens much later than when the GA+QL approach is used (see the last two columns in Table 2: at generation 100, travel times computed by the GA are higher than

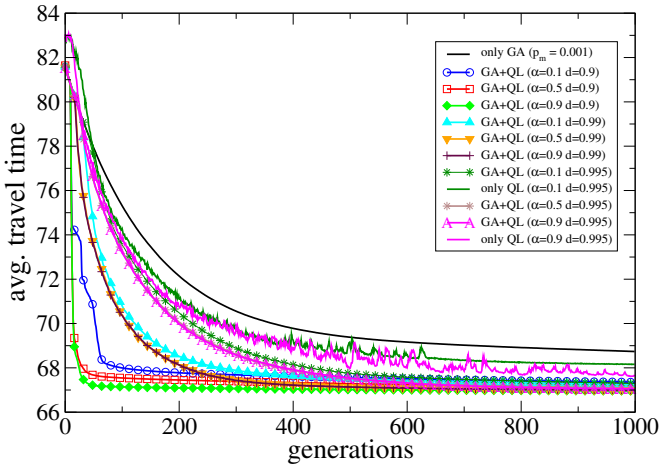

Fig. 4. Average travel time along generations ($k = 8$, $p_m = 0.001$, $c = 0.2$), for different situations: only GA, only RL (different parameter values), GA+RL (different parameter values regarding QL).

Table 2. Average Travel Time per OD Pair: comparison all-or-nothing, incremental assignment, GA-only, and GA+QL ($k = 8$, $p_m = 0.001$, $c = 0.2$, $\alpha = 0.9$, $d = 0.9$)

OD	All-or-nothing	Incremental	GA (gen. 100)	GA+QL (gen. 100)
AL	114	75.92	78.67	70.86
AM	94	70.18	71.27	65.22
BL	98	77.96	82.43	69.58
BM	71	62.48	69.21	62.42
all	96.35	71.77	75.31	67.14

those computed by GA+QL). Results referring to the use of Q-learning alone are good in many cases, but there is an exploration phase where random actions are selected[3]. Therefore it takes more time for the Q-learning alone to reach the same performance of the GA+QL approach.

As mentioned previously, not only the average travel time over all drivers matters, but also travel times in the four OD pairs. Table 2 (last column) shows these values by generation 100, when convergence has been achieved (in case of $\alpha = 0.9$ and $d = 0.9$).

Finally, because the minimization of the average travel time may not be fair for everyone, we have run the same approach in which the objective function is

[3] While this is also the case in the GA, we remark that GA is able to evaluate many solutions in parallel, which is not the case in the Q-learning.

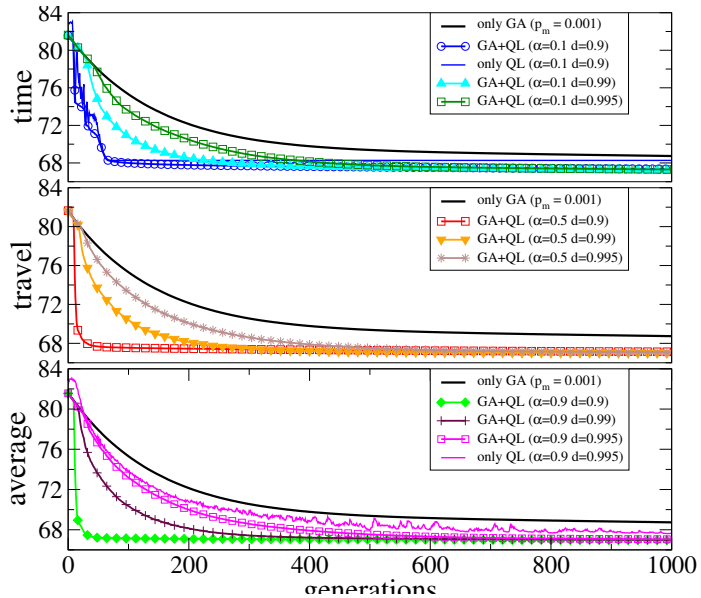

Fig. 5. Average travel time along generations ($k = 8$, $p_m = 0.001$, $c = 0.2$), for different situations: only GA, only RL (different parameter values), GA+RL (different parameter values regarding QL). The upper plot is for $\alpha = 0.1$, the lower for $\alpha = 0.9$, and the middle plot is for $\alpha = 0.5$.

to minimize: (a) the travel time of the worst trip; (b) the sum of the worst four travel times (one for each OD pair). These results appear in figures 6 and 7 respectively. Three main conclusions here are: first, the same pattern as before arises, namely, GA+QL accelerates the convergence regarding the pure GA: just compare orange to black lines in these plots. Second, the worst travel time(s) decrease(s) faster when GA+QL is used. Third, the fact that the objective function is changed to a minimization of the worst(s) travel time(s) does not affect the overall efficiency: the average travel time remains around 67.

5 Conclusions and Future Work

When recommending routes for drivers, these can be aligned with the assignment that approximates the SO. However, drivers still perform adaptive route choice at individual level, seeking the UE. To address these combination that happens in the real world, our approach combines GA with Q-learning, where routes that are learned at individual level accelerate the convergence of the GA. Our results show that this hybrid approach is able to find solutions (in terms of average travel time) that are better than other methods, and that this is done faster. Future works relate to simulating the learning process of heterogeneous trips, as for instance those associated with different learning paces by their drivers and/or

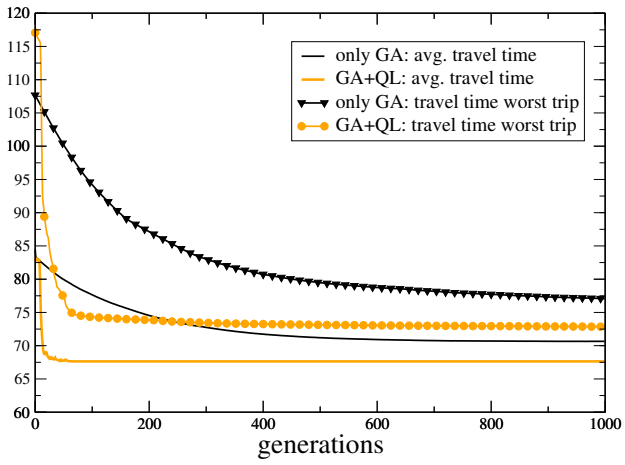

Fig. 6. Performance along generations: average travel time (over all trips – full lines – and for the worst trip – circles and triangles) when the objective function is to minimize the worst travel time.

adherence to route recommendation. This is barely addressed in the literature but is important because the population of drivers is heterogeneous.

References

1. Wardrop, J.G.: Some theoretical aspects of road traffic research. In: Proceedings of the Institute of Civil Engineers. Volume 2. (1952) 325–378
2. Ortúzar, J., Willumsen, L.G.: Modelling Transport. 3rd edn. John Wiley & Sons (2001)
3. Koutsoupias, E., Papadimitriou, C.: Worst-case equilibria. In: Proceedings of the 16th annual conference on Theoretical aspects of computer science (STACS), Berlin, Heidelberg, Springer-Verlag (1999) 404–413
4. Roughgarden, T., Tardos, É.: How bad is selfish routing? J. ACM **49** (2002) 236–259
5. Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V.V.: Algorithmic Game Theory. Cambridge University Press, New York, NY, USA (2007)
6. Gawron, C.: Simulation-based traffic assignment. PhD thesis, University of Cologne, Cologne, Germany (1998)
7. Buriol, L.S., Hirsh, M.J., Pardalos, P.M., Querido, T., Resende, M.G., Ritt, M.: A biased random-key genetic algorithm for road congestion minimization. Optimization Letters **4** (2010) 619–633
8. Wolpert, D.H., Sill, J., Tumer, K.: Reinforcement learning in distributed domains: Beyond team games. In: Int. Joint Conf. On Artificial Intelligence (IJCAI). (2001) 819–824

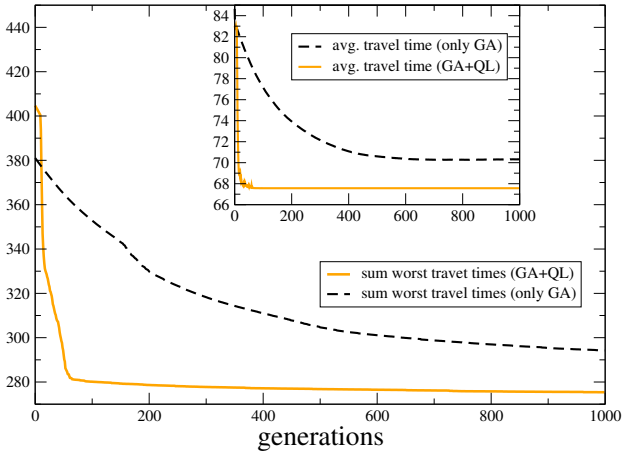

Fig. 7. Performance along generations. Main plot: value of the fitness function (sum over four worst travel times). Inset plot: average travel time over all trips.

9. Lima Júnior, F.d., Neto, A., Melo, J.d.: Hybrid Metaheuristics Using Reinforcement Learning Applied to Salesman Traveling Problem. INTECH Open Access Publisher (2010)
10. Henn, V.: Fuzzy route choice model for traffic assignment. Fuzzy Sets and Systems **116** (2000) 77–101
11. Cagara, D., Bazzan, A.L.C., Scheuermann, B.: Getting you faster to work: A genetic algorithm approach to the traffic assignment problem. In: Proceedings of the 16th Annual Conference on Genetic and Evolutionary Computation (companion). GECCO '14, ACM (2014) 105–106
12. Balmer, M., Rieser, M., Meister, K., Charypar, D., Lefebre, N., Nagel, K.: MATSim-T: Architecture and simulation times. In Bazzan, A.L., Klügl, F., eds.: Multi-Agent Systems for Traffic and Transportation Engineering. IGI Global, Hershey, US (2009) 57–78
13. Klügl, F., Bazzan, A.L.C.: Route decision behaviour in a commuting scenario. Journal of Artificial Societies and Social Simulation **7** (2004)
14. Dia, H., Panwai, S.: Intelligent Transport Systems: Neural Agent (Neugent) Models of Driver Behaviour. LAP Lambert Academic Publishing (2014)
15. Bazzan, A.L.C., Klügl, F.: Re-routing agents in an abstract traffic scenario. In Zaverucha, G., da Costa, A.L., eds.: Advances in artificial intelligence. Number 5249 in Lecture Notes in Artificial Intelligence, Berlin, Springer-Verlag (2008) 63–72
16. Yen, J.Y.: Finding the k shortest loopless paths in a network. Management Science **17** (1971) 712–716
17. Claus, C., Boutilier, C.: The dynamics of reinforcement learning in cooperative multiagent systems. In: Proceedings of the Fifteenth National Conference on Artificial Intelligence. (1998) 746–752