thus their actions have direct and often unwanted influences on others. For instance, the attempt to create a high-difficulty eco-circumstance for one user driver may accidentally create a high level challenge for another user driver (following behind the first user), who should instead experience a low-difficulty eco-circumstance; (2) user drivers are non-controllable entities that might also create unwanted eco-circumstances for other users; (3) the solution has to be found in real-time.

In this paper, we will represent our problem as a distributed constraint optimization problem (DCOP) [14]. In DCOP applications, each agent holds a variable and can change its value to achieve a globally optimal solution. To our knowledge, DCOP algorithms have not yet been applied to scenarios that are comparable to the heterogeneity of our multi-user multi-opponent real-time eco-driving scenario. Hence, the main contribution of this paper is a new application for DCOP algorithms and the presentation of results from a series of simulation runs that involve human users, a kind of "participatory simulation" [11].

The remainder of the paper is organized as follows. In Section 2, we discuss related work. Section 3 describes the core technical contribution of our paper: (1) offline Q-learning, (2) online challenge balancing, and (3) the translation of our multiuser, multi-agent application scenario to a DCOP. Section 4 presents the experiment and its results. Section 5 concludes the paper.

2. RELATED WORK

This section reviews related work on existing applications of multi-agent systems for traffic control problems, as well as on real-time challenge balancing systems.

2.1 Multi-Agent Systems in Traffic Control

Although our use of controllable agents in traffic is completely different from their use in 'usual' traffic control problems, we will discuss applications of multi-agent systems in traffic control to highlight similarities and differences.

The control of traffic signals is typically based on pre-established traffic signal plans that are designed to work with given nominal volume of vehicles. As this approach cannot cope with changes in these volumes [13], there is a need for approaches that allow for dynamic synchronization of adjacent lights. An approach based on swarm intelligence is proposed in [17]. Traffic signal agents see signal plans as tasks to be executed. They receive stimuli to change the tasks (i.e. chose a different plan) based on the amount of "pheromone" in each direction. This pheromone is continuously produced by vehicles waiting at the red light. Although traffic signals are able to achieve synchronization, the time needed to converge can be high. This approach is not suitable for highly dynamic environments where decisions have to be made in real time.

An agent-based mechanism for intersection control that minimizes fuel consumption is proposed in [19]. The work uses a reservation-based protocol directed towards autonomous guided vehicles (AGV), such as in [8]. Intersections use an emission model to estimate the increase in emissions, and assign time slots to vehicles in order to minimize emissions. Such an approach could be applied to our case, if we reverse the goal and assign time slots to maximize (rather than minimize) the emissions. However, our application also involves human users and there is no guarantee that they will respect the assigned time slot, as the AGVs do.

A DCOP-based approach to coordinate traffic signals is proposed in [16]. In this scenario, each agent is assigned a variable which indicates the direction of synchronization of a certain intersection. Adjacency between intersections represents a constraint between their variables, and the cost is determined by whether they are synchronizing in the direction with more incoming vehicles or not. In trying to solve the DCOP, agents eventually reach a configuration of minimum cost. When the traffic pattern changes the cost increases and the DCOP algorithm is automatically reactivated. Our model is also based on dynamic DCOPs. In our case, however, the topology of the constraint graph is also dynamic, since it depends on the position of each entity (user driver or moving opponent) on the road at a certain moment. Furthermore, as mentioned above, we also account for nodes that we cannot control, representing users.

2.2 Real-time Challenge Balancing

Real-time challenge balancing [15], a.k.a. dynamic difficulty adjustment [10], consists in adapting the challenge level of the system to the user's skill level in real time. The concept is based on Csikszentmihalyi's flow theory [6], which describes the state of "flow", a psychological state of full engagement and intrinsic motivation of performing the task at hand. This state can only be achieved if there is a balance between the challenge of the task and the skill of the performer. There is evidence that by keeping the challenge level and the user skill in equilibrium, users will be motivated to continue the task for longer periods of time, thus making training more effective.

In applications of RCB, the challenge is usually measured by a heuristic function that is called "challenge function". There is no general model for the challenge function, since it is application-dependent. By keeping track of this function, an application is able to detect when the challenge leaves a desirable range and can decide when to intervene.

Existing RCB methods include behavior-based approaches [22], neuro-evolutionary approaches [15], and an approach based on "challenge sensitive action selection" [1]. In the last mentioned approach, an agent is trained offline by standard tabular Q-learning [25] to learn a winning strategy against the user. When online, the system keeps track of a "level" n, which represents the user's skill level. By periodically analyzing the challenge function, the level n can be increased or decreased if the challenge is too low or too high, respectively. The agent then deliberately chooses not to perform the action with the best Q-value, but the nth best action for a given state, in order to match the user's skill. In our approach, the same method is used to train the "opponents".

It is important to note that all the RCB methods mentioned above focus on "one opponent vs. one user" scenarios, where one agent is an opponent that has to adapt to a user's performance. By contrast, our eco-driving scenario includes multiple users and multiple opponents, all sharing the same action (driving) space. Hence we have to deal with situations where we want to calibrate challenges for users with different skills in the same area. A traffic light cannot simultaneously turn "red" for one player and stay "green" for another. Hence, there is a need for an approach that

considers all entities involved, users and opponents, and tries to find a set of actions that optimizes challenges for all users.

3. MULTIUSER RCB WITH DISTRIBUTED CONSTRAINT OPTIMIZATION

In this section, we describe our application scenario and the application of DCOP methods to achieve the best possible challenge level for multiple users in real time. Following Andrade [1], we differentiate the *offline learning* phase from the *online adaptation* phase. Importantly, our online adaptation phase extends previous work on challenge balancing to the multi-user case, by using distributed constraint optimization.

3.1 The iCO$_2$ Application

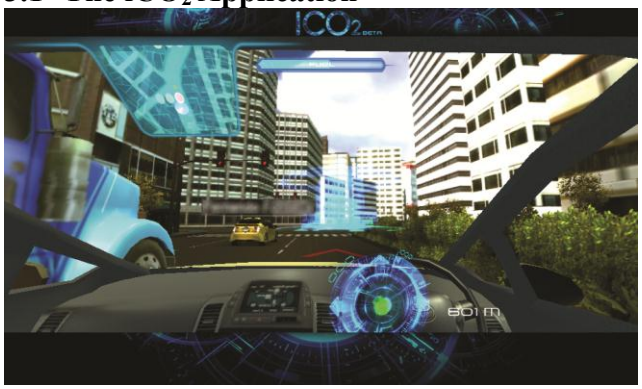

Figure 1. A screenshot of the iCO2 application on Facebook. The blue "wheel" (mid-right) is the driving interface. It shows "green" to indicate eco-friendly driving behavior

iCO$_2$ is an online tool for practicing eco-safe driving with multiple networked users. In September 2012, iCO$_2$ was launched as an application on Facebook.[1] The 3D scenario depicts an area in Tokyo of about 1 km^2. Users can control the direction and speed of the car operating a mouse-based interface (see Figure 1). The interface will change its color to indicate eco-friendliness in a continuous color range from green (eco-friendly) to red (eco-unfriendly), based on the EMIT emission model [4]. As a first approximation, eco-friendliness is defined by smooth acceleration and deceleration only.

iCO$_2$ is designed as a computer game, where the user (or player) tries to drive as far as possible with a given amount of fuel. To receive more fuel, the user has to pass "checkpoints" which are spread around the city. The distance between checkpoints increases and the amount of fuel each checkpoint yields decreases. Colliding with other cars or running a red light will result in a fuel penalty.

As a tool for eco-driving practice, the iCO$_2$ application has two unique features: (1) unlike existing single-user tools from major car makers, our application supports multiple drivers in a shared space, just like in a real traffic situation; (2) the application can dynamically create situations (eco-circumstances) that make eco-driving driving difficult. Eco-circumstances are created by two types of computer-controlled agents ("opponents"): traffic signals (opponents) and opponent cars.

The traffic signal perceives oncoming cars and decides when to turn to "red". The goal of this opponent is to generate the "yellow-light dilemma", where the driver does not know whether to brake or to accelerate and cannot avoid choosing an eco-unfriendly action [21]. The opponent car perceives the car behind it and decides whether to accelerate, to brake or to keep the speed. The goal of this opponent is to induce an eco-unfriendly behavior on the immediately following car, by suddenly braking hardly and accelerating hardly. The opponent car is controlled by the intelligent driver model (IDM) [23]. Braking and accelerating behavior is achieved by adjusting parameters in the IDM.

3.2 Offline Learning of Q-Values

To avoid situations where the behavior of opponents (traffic signals and opponent cars) makes eco-driving for the user too easy or too difficult, we apply real-time challenge balancing to the opponents' actions [15].

In the offline learning phase, we train our opponents to achieve challenge balancing against one user. Using Q-learning, the opponents learn an optimal policy to make eco-driving difficult for the user. The learning phase generates two q-tables, one for the traffic signal and one for the opponent car. Note that there is no learning after the tables are generated.

3.3 Online Adaption based on the Challenge Function

In the online adaptation phase, an opponent can decide not to choose the optimal action (provided by the Q-table), but select progressively less and less suboptimal actions until one matches the user's challenge requirements. A challenge function informs the decision by evaluating the average challenge faced by a user in real-time, using the following heuristic:

$$c_t = \left(fuel * w_f + accel^2 * w_a + RED * w_{red} + COL * w_{col}\right)$$

Here c_t is the instantaneous challenge at a time t; $fuel$ is the fuel consumption, calculated by the EMIT model; $accel$ is the acceleration of the car, squared because high and low values of the acceleration should be avoided; RED is the rate of crossing red traffic lights; and COL is the rate of collisions with other vehicles. The constants w_f, w_a, w_{red} and w_{col} represent the weights for each component in the equation. The instantaneous challenge is calculated at a fixed sampling rate of 300ms, and accumulated according to a linear interpolation:

$$AC_t = c_t * \alpha + c_{t-1} * (1 - \alpha)$$

Here AC_t is the accumulated challenge at a time t. At each second, the current accumulated challenge is added to a moving average of size 10. The average challenge is the result of the moving average (i.e. it will always contain the accumulated challenge of the previous 10 seconds). At a rate of 10 seconds, the average challenge is evaluated and, if it crosses a threshold, a sub-optimal action is chosen. For instance, if the challenge is too high, the opponent will start to choose the action with the 2nd-best Q-value, and if the average challenge continues high at the next evaluation point, the 3rd-best Q-value is selected, and so on. This technique is called "challenge-sensitive action selection".

[1] http://www.facebook.com/icotwo/

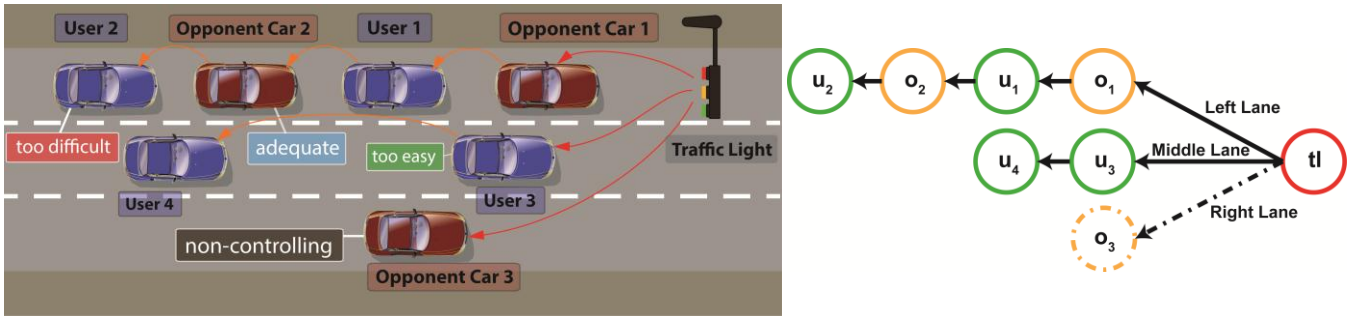

Figure 2. Left: Example traffic situation involving traffic signal, user driven cars (labeled with challenge requirements) and opponent cars. Right: DCOP graph structure corresponding to the situation on the left. The root is always a traffic signal (tl), and each branch represents a lane of the street. Constraints represent dominance relationships. All leaf nodes are user agents (u_i). o_3 is not considered in the computation as it is non-controlling

During the online phase, all opponents use the two tables to look up an action policy for individual users. When multiple users share the same area, a more advanced method is required, as we have to find an optimal set of actions for multiple opponents to execute, so that challenge is optimal for all users involved. Such a method will be described in the next section.

3.4 Multiuser Challenge Adaptation based on Distributed Constraint Optimization

In distributed constraint optimization problems (DCOP), each agent is assigned to one or more variables and these have interdependencies. The goal is to find an optimal assignment for the variables to minimize or maximize a global cost function [7].

A DCOP is formally defined by (following the notation of [7]):

- A set of *N agents,* $A = \{A_1, A_2, \dots, A_N\}$, whereby an agent is either computer-controlled (an opponent) or user-controlled.
- A set of *n variables,* $V = \{x_1, x_2, \dots, x_n\}$. In our case, each agent holds a single variable, hence $n = N$.
- A set of *domains* $D = \{D_1, D_2, \dots, D_n\}$, where the value of x_i is taken from D_i. Each D_i is assumed to be finite and discrete.
- A set of *cost functions* $f = \{f_1, \dots, f_k\}$ where each f_i is a function $f_i : D_{i,1} \times \dots \times D_{i,j} \rightarrow N \cup \infty$. Cost functions can also be called *constraints*.
- A *distribution mapping* $Q: V \rightarrow A$ assigning each variable to an agent. $Q(x_i) = A_i$ means that A_i is responsible for choosing a value for x_i. A_i is given knowledge of x_i, D_i, and all f_i involving x_i.
- A *global cost function* F defined as an aggregation (e.g. summation) over the set of cost functions.
- An *assignment* $\{d_1, \dots, d_n\}$ representing the values of each variable x_i, taken from their domains D_i.

A DCOP is represented by a constraint graph, where the nodes are the variables and a link between two nodes exists when there is a constraint over two variables. Agents whose variables share a constraint are called "neighbors". Agents are only allowed to see their neighbors, and may exchange messages with them.

In our application, each agent represents either an opponent or a user. In the case of an opponent, the variable associated with it indicates the action that the opponent will choose to execute. Since the users are non-controllable entities, the variable indicates the action that we predict the user to execute. The prediction is based on (1) the current environment state (car's position, acceleration, speed, and distance to the light) and (2) the actions that the car ahead may perform (i.e. the value of the parent node). It is important to note that not all opponents involved in the DCOP may influence the prediction of a certain user's action. To reflect this, we introduce the concept of "dominance". An agent (opponent or user) is dominant over another agent (opponent or user) if the actions of the former directly influence the behavior of the latter. In our application:

1. A traffic signal is dominant over an agent A_i, if A_i is in front of the traffic light and there is no other agent A_j in the environment between A_i and the traffic light.
2. An opponent/user car is dominant over an agent A_i, if A_i is behind the opponent/user car, in the same lane, and there is no other agent A_j in the environment between A_i and the opponent/user car.

An opponent is called *non-controlling* if it is not dominant over any other agent (at a particular moment). Non-controlling opponents are not part of the constraint graph, since they cannot influence any user. In our scenario, DCOPs always involve a traffic light. Hence, we obtain a tree-like structure, where the traffic light is the root, each branch represents a lane, and the leaf nodes are users (see Figure 2 (right)). This means that if agent A_i is dominant over agent A_j, the node controlled by A_i is a parent node of the node controlled by A_j. Hereafter, we refer to the nodes using tree notation (parent node and child node). Once the structure is determined from the position of the traffic lights, opponent cars and users in the 3D environment, we can start with distributed constraint optimization.

The domain of a variable in the DCOP depends on the type of the agent controlling that variable. If the variable corresponds to a traffic signal node, there are two available actions, "change color" and "not change color": D = {C, NC}. If the variable corresponds to an opponent car node, the available actions are braking, keeping the speed and accelerating: D = {B, KS, A}. In the case

of the user node, the domain is the output of the prediction function, which is also "braking", "keeping speed", or "accelerating": D = {B, KS, A}. Even though the user is allowed to change lanes, this is not covered by the prediction function.

The existence of constraints is solely based on the dominance relationship between agents. The cost of existing constraints depends on whether there is a conflict between the agent's actions or not. A conflict occurs when (a) a parent node (dominant agent) executes an action that raises an inadequate challenge for the child node (in case it is a user node); (b) a child node executes an action that is conflicting with its parent node's value. As an example of the latter type of conflict, consider an opponent car that is in front of a traffic light and decides to accelerate, while the light decides to change to red. If the car is very close to the light, it may be able to accelerate and cross in time; hence both actions can be executed independently. However, if the car is slightly further away, it may be forced to brake, even if it had chosen to accelerate (recall that opponents never cross a red light). This unwanted braking will affect the following cars and hence, this kind of conflict should be detected beforehand.

The cost of each constraint between two agents, $f(x_i,x_j)$, is computed according to the Algorithm 1.

Algorithm 1: Calculates the cost $f(x_i, x_j)$, where x_i is parent of x_j

if (x_j is opponent car) **then**

  $predicted_j \leftarrow PredictAction(x_j, x_i, d_i)$

  if ($predicted_j > x_j$) **then** *// $predicted_j$ is less restrictive than x_j*

    $f(x_i, x_j) \leftarrow 0$

  else

    $f(x_i, x_j) \leftarrow \infty$

  end if

else

  $predicted_j, chal_j \leftarrow PredictChallengeAndAction(x_j, d_i)$

  if ($predicted_j$!= x_j)

    $f(x_i, x_j) \leftarrow \infty$

  else

    $f(x_i, x_j) \leftarrow |(\text{max challenge} - \text{avgChallenge}(x_j)) - (chal_j)|$

end if

Case 1: When the child node x_j is an opponent car, we predict the action of the opponent car given the value of the parent x_i, and the current state. Then we compare the predicted action with the desired action (d_j, the value of x_j). This comparison is based on the restrictiveness of the action, "brake" being the most restrictive action and "accelerate" being the least restrictive one. If the predicted action is more restrictive than the desired one, they are conflicting, and the constraint has infinite cost. Otherwise, there is no cost ($f(x_i, x_j) = 0$).

Case 2: When the child node x_j is a user, we similarly predict the action that the user car will execute, given the value of the parent x_i and the current state. Furthermore, we predict the challenge that this action represents by using the challenge function. If the predicted action is different from the value of the child node x_j, there is infinite cost. This is necessary because the DCOP algorithm investigates all the possible values for each node. Since the user node should always represent a predicted action, values that are inconsistent with it should yield infinite cost. If the predicted action is the same as the value of the child node x_j, the cost is the difference between the predicted challenge and the ideal challenge for that user. The ideal challenge for a user is the difference between the maximum challenge and the current average challenge for the user.

For the prediction of the actions of opponent car and the user car, we use the Intelligent Driver Model (IDM) [23]. The IDM is a parameterized equation that calculates the ideal acceleration, based on the headway (distance to the front vehicle) and speed of the front vehicle. Hence the prediction is based on headway and speed. Another parameter is the "prediction time", which indicates how far ahead the function is looking. If the parent node is a traffic signal, we use the IDM with Intersection Management (IDM-IM) [9], which is the IDM with additional capabilities to indicate how a car behaves at an intersection. If the parent is a car, the usual IDM is used.

The acceleration output of the IDM is the used to determine:

- The predicted action: negative accelerations are mapped to the "brake" action, positive to the "accelerate" action, and around zero (given a threshold) to the "keep speed" action
- The challenge (in case of a user node): using the predicted acceleration as input to the emission model and to the challenge function

Algorithm 2: Calculates the predicted action PredictAction

PredictAction(x_j, x_i, d_i)

predSpeed $\leftarrow$ speed(x_i) + acceleration(x_i)*predictTime

myPredSpeed $\leftarrow$ speed(x_j) + acceleration(x_j)*predictTime

predPosition $\leftarrow$ position(x_i) + speed(x_i)*predictTime + acceleration(x_i)*predictTime^2/2

myPredPosition $\leftarrow$ position(x_j) + speed(x_j)*predictTime + acceleration(x_j)*predictTime^2/2

predAcceleration $\leftarrow$ IDM(myPredSpeed – predSpeed, distance(myPredPosition, predPosition))

if (|predAcceleration| < ε) **then**

  return keep speed

else if (predAcceleration > 0)

  return accelerate

else

  return brake

In order to take advantage of the tree structure, we use the DTREE algorithm [18], a complete and linear-time algorithm for solving DCOPs that can be represented as a tree network. In DTREE, there is an initial step of selecting the root node, which is not implemented in our case, since our application has a unique root node (the traffic signal). When the algorithm starts, leaf nodes calculate the lowest cost they can achieve for each of their parent's values, and send a UTIL message to their parent, informing the lowest cost table. Parents aggregate the values received from their children to find out, for each possible action, the best utility they can achieve. When the messages reach the root node, it is able to choose an assignment for itself and send it back to their children, as a VALUE message. Each child node,

then, chooses its own value, based on the pre-calculated utilities and its parent's assignment. The algorithm finishes when the VALUE messages reach the leaf nodes. After the DTREE is finished, each of the opponents involved in the DCOP executes its chosen action (node assignment) in the real world.

4. EXPERIMENT

To investigate the suitability of the proposed approach to our scenario, we conducted a pilot study involving human users. The purpose of the study was two-fold.

- Performance: we wanted to test the performance of the DCOP algorithm in a real-time situation with non-controllable agents (i.e. users).
- Scalability: we wanted to investigate the scalability of the algorithm, as our scenario allows having multiple users and computer-controlled entities simultaneously.

Regarding performance, we hypothesized that a linear-time algorithm as the DCOP calculation would not be a bottleneck for the application. As to scalability, we speculated that if the number of agents (users and opponents) grows, either the size of the graphs, or the frequency of the calculations of DCOPs will increase to a point where the number of agents becomes a bottleneck to finding the optimal solution.

Furthermore, we wanted to assess how the challenge function for users develops over time. This can provide us an insight into the quality of the current challenge function and into how events inside the environment (e.g. participation in DCOPs, collisions, stopping at crossings, etc.) relate to the calculated challenge.

4.1 Method

4.1.1 Subjects and Design

We had 30 subjects (19 male, 11 female), on average 27.6 of age. Most subjects were students at institution. Subjects were paid an amount equivalent to USD 10 for participation.

Since it is difficult to have a large number of users driving at the same time in a controlled experiment, we also used computer-controlled cars to represent users in the scenario. We call such car "IDM-Eco-Car", because it is driven by the IDM, but parameterized to drive in an eco-friendly way. Traffic signals or opponent cars do not differentiate between IDM-Eco-Cars and human users. The IDM-Eco-Car is currently a limited representation of the user, as e.g., it cannot change lanes. Opponent cars are not allowed to change lanes. Both the opponent cars and the IDM-Eco-Cars always respect the red light and never collide. Finally, human users are allowed to drive freely among the opponent cars and IDM-Eco-Cars.

4.1.2 Materials and Procedure

Subjects drove in a networked 3D simulation space representing Yasukuni Street, a major street in Tokyo (see Fig. 1). For the purpose of the study, the road was made a closed circuit (no turns, no cars coming in the opposite direction). Traffic signals are distributed along the road. To increase unpredictability of the scenario some traffic signals are opponents, others stay green all the time. Groups of opponent cars are created at distributed spots along the road, usually close to a traffic signal. This was done with the intention of increasing the probability of having users and opponents in front of a traffic signal at the same time.

Each subject was assigned to a computer with a mouse, a monitor and headphones. Subjects were positioned so that they could not see each other's screen. Subjects drove in groups of three at a time (10 groups in total), and were instructed to try to accelerate smoothly and to avoid sharp braking. They were told to respect the traffic rules (avoid collisions and crossing the red light) and drive with the speed of up to 60 km/h.

The experiment was divided into one practice session and three experimental sessions, where the number of computer-controlled cars was increased in a controlled manner. In the practice session, three users practice driving together for five minutes; there are only users and traffic light opponents.

In the experimental sessions, three users drive around the circuit together for 10 minutes.

- Session I: nine opponent cars, divided into three groups of three cars, are created along the road.
- Session II: 18 opponent cars, divided in 3 groups of 6, plus 3 IDM-Eco-Cars.
- Session III: 27 opponent cars, divided in 3 groups of 9, plus 6 IDM-Eco-Cars.

Each time a DCOP is solved, the agents involved, graph structure, time taken and final assignment is logged. We also logged driving behavior data, such as position, speed, acceleration, fuel consumption, emission, etc., at a constant sampling rate of 100ms.

4.2 Results

The results of our pilot study are summarized in Table 1. The traffic signal is included in the size of the DCOP (e.g. a problem of size 7 should have one signal and 6 cars). As expected, when the number of entities in the scenario grows, the number of DCOPs to be solved also grows, as well as the size of the DCOP. Even though 33 computer-controlled cars and 3 human users shared the same circuit in Session III, the DCOP maximum size did not exceed 11 variables (traffic light included).

Table 1. Results per session

	Session I	Session II	Session III
Number of opponent cars	9	18	27
Number of IDM-Eco-Cars	0	3	6
Average time per DCOP	1.026s	1.004s	1.012s
Maximum time	2.089s	2.076s	2.162s
Number of DCOPS solved	127	449	750
Average size of DCOPs	4.496	4.897	5.352
Maximum size	7	9	11
Graph reconstructs (lane changing)	1	2	3

The study showed that regardless of the growth in size, the average time taken to solve the DCOPs remained approximately constant (1 second). Even if 1 second appears slow for graphs of such small size, it is important to note that this value does not represent solely the algorithm runtime. Since all of the graphics and logic complexity of the application runs in a single thread, the exchange of messages between agents has a natural delay, even if it runs locally. Hence, the actual time consumed by the DCOP algorithm might be much smaller than 1 second.

As a metric for the level of "dynamics" (rate of change of the position of user cars) of the traffic situation, we counted the number of reconstructions of the DCOP graph due to lane changing. First, the graph is always reconstructed when a car crosses the traffic signal (i.e. leaves the graph). It would also be reconstructed if a vehicle enters the relevant area. However, such situation was not foreseen in our experiment. Second, the lane changing action induces a reorganization of the existing variables and indicates how "dynamic" did the user cars behave, when inside the DCOP area. However, while users were allowed to freely move within the scenario, few graph reconstructions were detected in each session (see Table 1).

Although challenge balancing was not the focus of this study, we also looked at whether the DCOP calculations have any effect on the value of the challenge function over time. Fig. 3 shows the example for one user in Session II. For this experiment, the challenge function was weighted so that it could assume values within the range [0-4]. The peak at 2min50s indicates a vehicle collision (challenge value raises because of the collision itself, and the sudden deceleration generated by it). The figure shows that, overall, the challenge for this user remained low throughout the whole session. This could indicate many things: opponents' behavior is not aggressive enough, user was not affected by challenge balancing (as this would have yielded a generally higher challenge, in average) or that the challenge function needs to be adjusted. Further investigation is necessary to establish a direct relation between the participation in the DCOPs and the change in the value of the challenge function.

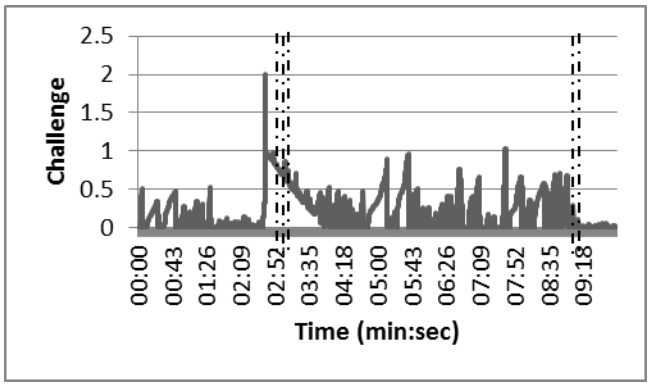

Figure 3. Challenge function over time, for one certain user in Session II. Vertical dotted lines represent DCOPs in which this user participated.

5. DISCUSSION AND CONCLUSIONS

The aim of this paper is to investigate multi-agent scenarios involving controllable agents and non-controllable agents (users) for the purpose of letting users practice eco-driving in an engaging and motivating way. This is achieved by adjusting the difficulty level for each of them. This aim led to the creation of iCO2, a Facebook application based on our multiuser networked 3D virtual environment technology. Unlike existing online training environments, which support a single user driver only, drivers in iCO2 share the same simulation space. Further, by introducing "opponents" (for users) to the environment, we try to create situations for user that make it difficult to drive in an eco-friendly way.

The research challenge is to find an adequate challenge level for all users (drivers) in spite of the high interdependency of a traffic situation consisting of multiple agents, including computer-controlled cars and users. To address this problem, we cast the scenario as a distributed constraint optimization problem (DCOP), and apply the DTREE algorithm [18] to find an optimal solution.

To test the algorihm on our scenario, we conducted a pilot study involving human drivers. The study revealed that the algorithm does not represent a bottleneck for the application and is suitable for a real-time application. However, more stress tests are necessary, since increasing the number of computer-controlled cars in the scenario did not raise the size of the DCOPs as much as expected. This might be due to the fact that our experiment involved 10s rather than 100s of computer-controlled vehicles besides the user drivers. To increase the amount of realistic "ambient traffic" (IDM-Eco-Cars), we have to increase the varibility of parameter settings of the IDM and implement further driving behaviors, such as lane changing. We expect that traffic with hundreds of cars will create more challenge for the DCOP algorithm, and interesting trade-offs between different algorithms might be observed.

In addition we want to increase the number of simultaneous user drivers in the driving simuation space. For that purpose, developed iCO_2, a Facebook application that supports massively multiuser driving simulation (see the link in Footnote 1). As a next step, we will organize an eco-driving 'contest' on this popular social networking service, which will allow us to gather large numbers of users in the scenario. This might lead to large-scale user study, which is hard to conduct in a laboratory environment.

Further, the granularity of the behavior (learned by Q-learning) of the opponent cars is rather coarse. We consider to retrain the car opponents with larger set of behaviors, with different levels of acceleration and braking.

In conclusion, we hope that our eco-driving training scenario can provide an innovative and challenging environment for testing DCOP algorithms. To the best of our knowledge, existing algorithms have not yet been applied to such potentially highly dynamic application environments involving heterogeneous types of agents.

6. ACKNOWLEDGEMENTS

This research was partly supported by a ''Grand Challenge'' Grant from National Institute of Informatics (NII), Tokyo and by the MEXT (Ministry of Education, Culture, Sports, Science, and Technology - Japan) scholarship applied for graduate studies at NII. The authors would like to thank all colleagues at NII for helpful discussions.

7. REFERENCES

[1] Andrade, G., Ramalho, G., Santana, H., and Corruble, V. 2005. Extending reinforcement learning to provide dynamic game balancing. In Proc. of the Workshop on Reasoning,

Representation, and Learning in Computer Games, 19th International Joint Conference on Artificial Intelligence. IJCAI. 7–12.

[2] Bazzan, A. L. 2005. A distributed approach for coordination of traffic signal agents. Autonomous Agents and Multi-Agent Systems 10, 131-164.

[3] Bazzan, A. L. C. 2008. Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. Autonomous Agents and Multi-Agent Systems, 18, 3, 342–375.

[4] Cappiello, A., Chabini, I., Nam, E.K., Lue, A. and Abou Zeid, M. 2002. A statistical model of vehicle emissions and fuel consumption. In Proc. of the IEEE 5th International Conference on Intelligent Transportation Systems. 801- 809.

[5] Cowley, B., Charles, D., Black, M., and Hickey, R. 2008. Toward an understanding of flow in video games. Computers in Entertainment, 6, 2 (Jun. 2008), 20:1-20:27.

[6] Csikszentmihalyi, M. 1990. Flow: The Psychology of Optimal Experience. Harper Perennial, New York.

[7] Davin, J. and Modi, P. J. 2005. Impact of problem centralization in distributed constraint optimization algorithms. In Proc. of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'05), 1057-1063.

[8] Dresner, K.,and Stone, P. 2005.Multiagent traffic management: An improved intersection control mechanism. In Proc. 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'05), 471-477.

[9] Fiore, M., Harri, J., Filali, F. and Bonnet, C. 2007. Vehicular mobility simulation for VANETs. 40th Annual Simulation Symposium, 301-309.

[10] Hunicke, R., and Chapman, V. 2004. AI for dynamic difficulty adjustment in games. Challenges in Game Artificial Intelligence AAAI Workshop. 91–96.

[11] Ishida T., Nakajima Y., Murakami Y., Nakanishi H. 2007. Augmented experiment: Participatory design with multiagent simulation. In Proc. of the International Joint Conference on Artificial Intelligence, 1341-1346.

[12] Jain M., An B., and Tambe M. 2012. An overview of recent application trends at the AAMAS conference: Security, sustainability and safety. AI Magazine, to appear.

[13] Junges, R., and Bazzan, A. L. C. 2008. Evaluating the performance of DCOP algorithms in a real world, dynamic problem, In Proc. of 7th Int. Conference on Autonomous Agents and Multiagent Systems (AAMAS'08), 599-606.

[14] Liu, J., and Sycara, K. P. 1995. Exploiting problem structure for distributed constraint optimization. In Proc. of International Conference on Multi-Agent Systems, 246-253

[15] Olesen, J. K., Yannakakis, G. N., and Hallam, J. 2008. Real-time challenge balance in an RTS game using rtNEAT. In Proc. of IEEE Symposium on Computational Intelligence and Games. CIG'08. 87-94.

[16] Oliveira, D. D., Bazzan, A. L. C, and Lesser, V. 2005. Using cooperative mediation to coordinate traffic lights: A case study, In Proc. of the 4th International Conference on Autonomous Agents and Multiagent Systems, 463–470.

[17] Oliveira, D., Ferreira, P. R., Jr., Bazzan, A. L. C., and Klügl, F. 2004. A swarm-based approach for selection of signal plans in urban scenarios. In Proc. of 4th International Workshop on Ant Colony Optimization and Swarm Intelligence—ANTS 2004, 416–417.

[18] Petcu, A. and Faltings, B. 2004. A distributed, complete method for multi-agent constraint optimization. In CP 2004 - Fifth International Workshop on Distributed Constraint Reasoning,1-15.

[19] Pulter, N., Schepperle, H., and Böhm, K. 2011. How agents can help curbing fuel combustion–a performance study of intersection control for fuel-operated vehicles. In Proc. of 10th Int. Conference on Autonomous Agents and Multiagent Systems (AAMAS'11), 795–802.

[20] Rogers, A., Maleki, S., Ghosh, S. and Nicholas R, J. 2011. Adaptive home heating control through Gaussian process prediction and mathematical programming. In Proc. of the Workshop on Agent Technologies for Energy Systems, 71–78.

[21] Seifert, Howard S. 1962. The stop-light dilemma. American Journal of Physics, Volume 30, Issue 3, 216-218.

[22] Tan, C. H., Tan, K. C., and Tay, A. 2011. Dynamic game difficulty scaling using adaptive behavior-based AI. IEEE Trans. on Computational Intelligence and AI in Games *3*, 4, 289–301.

[23] Trieber M., Hennecke A. and Helbing D. 2000. Congested traffic states in empirical observations and microscopic simulations, Phys. Rev. E 62, Issue 2, 1805-1824

[24] Vandael, S., Craemer, K. D., Boucke, N., Holvoet, T., and Deconinck, G. 2011. Decentralized coordination of plug-in hybrid vehicles for imbalance reduction in a smart grid. In Proc. of the International Conference on Au- tonomous Agents and Multiagent Systems (AAMAS'11), 803– 810.

[25] Watkins, C. and Dayan, P. 1992. Q-Learning. Machine Learning, 8, 279-292.