"Let's Save Resources!": A Dynamic, Collaborative AI for a Multiplayer Environmental Awareness Game

Pedro Sequeira, Francisco S. Melo and Ana Paiva
INESC-ID and Instituto Superior Técnico, University of Lisbon
Av. Prof. Dr. Cavaco Silva, 2744-016 Porto Salvo, Portugal
pedro.sequeira@gaips.inesc-id.pt, fmelo@inesc-id.pt, ana.paiva@inesc-id.pt

***Abstract*—In this paper we present a collaborative artificial intelligence (AI) module for a turn-based, multiplayer, environmental awareness game. The game is a version of the EnerCities serious game, modified in the context of a European-Union project to support sequential plays of an emphatic robotic tutor interacting with two human players in a social and pedagogical manner. For that purpose, we created an AI module capable of informing the game-playing and pedagogical decision-making of the robotic tutor. Specifically, the module includes an action planner capable of, together with a game simulator, perform forward-planning according to player preferences and current game values. Such predicted values are also used as an alert system to inform the other players of near consequences of current behaviors and advise alternative, sustainable courses of action in the game. The module also incorporates a social component that continuously models the game preferences of each player and automatically adjusts the tutor's strategy so to follow the group's "action tendency". The proposed AI module is therefore used to inform about important aspects of the game state and also the human players actions. In this paper we overview the properties and complexity of this collaborative version of the game and detail the AI module and its components. We also report on the successes of using the proposed module for controlling the behavior of a robotic tutor in several experimental studies, including the interaction with children playing collaborative EnerCities.**

I. Introduction

The development of artificial tutors that can assist students in their learning activities and provide educators with a rich and multi-faceted tool for teaching has been the topic of significant research in recent decades. The EMOTE project (http://www.emote-project.eu/) aims at developing a new generation of artificial embodied tutors that have perceptive capabilities to engage in empathic interactions with learners in a shared physical space [1]. Towards this goal, a learning scenario involving a modified version of the serious game EnerCities (EC) [2] was developed in which an artificial tutor interacts with two students towards the construction of an energy-sustainable city.

The multiplayer version of EC featured in the learning scenario is a collaborative game, where each of the three players assumes one of three possible roles: *Mayor*, *Economist* and *Environmentalist*. Each player has access to a different set of actions that contribute to the development of the city in different ways. By jointly discussing the impact of different actions in the sustainability of the city, the students have the opportunity to get in touch and deepen their understanding about processes such as sustainability, greenhouse effect, acid rain, and global warming.

In light of the pedagogical goals of the project, the artificial tutor should not only act as one of the players, but also to interact with the other players, highlighting the impact of certain actions and assisting in their understanding of the long-term effects of different plays [1]. In its plays, the artificial tutor should be able to both induce and avert certain situations, in order to present the students with a variety of scenarios about which they can discuss the topics of the game. The ability of the tutor to "drive" (to some extent) the flow of the game must rely on a robust, flexible and adaptive artificial intelligence module that is able to plan the game play towards different goals that may be adjusted along the game [3].

In this paper we propose an artificial intelligence (AI) module that enables the artificial tutor in EMOTE to play the multiplayer version of EC in an effective and adaptive manner. The AI module is able to play optimally—in a score maximizing sense—while allowing a human supervisor to adjust the score that guides the action selection process. In this manner, the tutor's play can vary from more "environment-friendly" to completely "money-driven". Besides allowing for this flexibility, the AI module also maintains a high-level model of the strategy adopted by the other players, and automatically adjusts its own scoring process to maximize the probability of reaching a sustainable city configuration.

We propose a search-based approach in which the states (corresponding to the nodes in the search tree) capture relevant features of the game that can be weighted differently depending on the specific play-behavior that is desired from the tutor. At the same time, the algorithm keeps a statistic that summarizes play tendencies of the other players, which allows the search to adapt to the particular strategies adopted by the other players.[1]

Our results show that the AI module is able to effectively respond to different strategies by the other players. Additionally, the particular state representation adopted provides the necessary input to other modules of the artificial tutor, enabling it both to *explain* its own plays as well as *discuss* the advantages or disadvantages of the plays of others.

II. A Multiplayer, Collaborative EC Game

As mentioned above, given the collaborative and pedagogical nature of the interactive scenario developed in EMOTE, the original game of EC had to be modified to support the sequential plays between the two human subjects and

[1]This can be seen as a feature-based adaptation of *fictitious play* [4].

978-1-4799-8622-4/15/$31.00 ©2015 IEEE

						3		
		4	3	3	3	1	3	4
2		2	2	1	1	1	3	4
		2	1	1	4	1	3	4
2		1	4	City Hall	1	1	3	4
		2	1	1	1	2	3	4
2		4	2	2	2	3	4	

Fig. 1. A grid-world representation of the available cells in EC. Numbered cells represent the level at which they are available in the game. "City Hall" is a built-in structure. Different colors represent distinct *Surfaces*.

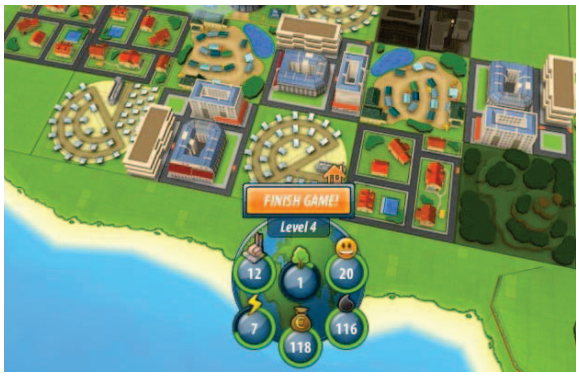

Fig. 2. The user interface of MCEC with all game scores, population and resources levels visible to the players in a unified location of the screen.

the autonomously controlled robotic tutor. In this section we overview the rules and dynamics behind EC and highlight the differences detail its *multiplayer*, *collaborative* version of the game, henceforth denoted by MCEC, for which we developed the AI module.

A. Game Rules and Dynamics

As explained earlier, EC is a turn-based game whose global objective is to raise awareness on environmental issues related with energy consumption for people playing the game [2]. Associated with EC is a *city* with a certain *population level*. In addition, the city has a set of resources whose levels vary throughout the game as a consequence of the player's actions, namely *oil*, *(electric) power* and *money*. The game also has a set of *scores* related to different aspects of the city's health, specifically *well-being*, *environment* and *economy*.[2] In terms of actions, at each turn, the player can *build* a new structure, *upgrade* up to 3 existing structures, *implement policies* or simply *skip* the turn. The goal of the EC game itself is to build a *sustainable city* until a certain level of population is attained. We refer to such goal as the *game designer's goal*.

The game's city is represented by 45 distinct *cells* on which a player can build structures. There are 21 different structures available and 56 distinct upgrades one can perform, thus averaging at 2.6 upgrades per structure. A grid-world view of the city and its cells is depicted in Fig. 1, including the *game level* at which they become available. Each structure and the upgrades performed over them, together with the type of *surface* on which it is built, dictate part of the cell's contribution to the different game scores as well as the change, in each turn, to the available resources and population. The other part is determined by influences from the adjacent structures and surfaces in the 4 cardinal directions. A game session progresses trough 4 different levels. A change in level occurs when a certain amount of population is reached, making available more kinds of structures and more space on which to build them, as can be seen in Fig. 1. The game ends whenever one these conditions is met: a *sustainable city* with a population level of 200 is reached in level 4; the city runs out of non-renewable resources, *i.e.*, the oil level goes below zero. A city is considered sustainable when all game scores have an above-zero level. In practice, the objective of a subject playing EC is to satisfy the first condition. We refer to such goal as the *players' goal*.

[2]In the original version of EC, this scores are respectively referred to as *people*, *planet* and *profit*. Due to the multiplayer nature of MCEC, we chose a different nomenclature.

B. Multiplayer EnerCities

The transition from EC to its multiplayer version, MCEC, is straightforward. The game can be played by up to three players, each portraying a different *role* related to one of the game scores. Specifically, a *Mayor* player is associated with the well-being score, an *Economist* player with the city's economy and an *Environmentalist* with the environment score. Apart from policy-implementation and actions that improve the city's population level (residence-building actions) and the city's energy level, all other available actions are directed at enhancing a specific game score. As a consequence, in MCEC each role-playing subject will have available only a subset of the game's build and upgrade actions, *i.e.*, the ones directed at improving its associated score. In the context of the scenario being developed, the interaction with the game is also transformed to support multiplayer interaction between the robotic tutor and the human subjects within the same physical space. This is achieved by displaying the game's interface in a large multi-touch table and letting the subjects perform actions by clicking in a separated area of the table's screen [5].

C. Challenges in MCEC

In this section we analyze the challenges in creating an AI module to control the behavior of a robotic tutor in a scenario involving interactions with human subjects playing MCEC. A challenging aspect of building an AI module for MCEC is that there is no "global score" to be maximized during the game. Instead, a series of conditions related to game scores and resources values has to be satisfied at each turn so that a city is sustainable. For example, to advance in the game one must build residential structures. As an effect, this will generally impact the resources being spent, *i.e.*, less money available, more energy and oil being consumed. In turn, this causes a need to address the shortfall resources by building other kinds of structures. The more of these structures are built, the less space is available to build homes to raise the population's volume and therefore advance through the levels of the game. Ultimately, in EC there are no "wild" or "magic"

actions that augment all the relevant game levels at the same time. Therefore, a *careful* consideration of all aspects and their future impact in the city's health must be taken into account at each turn throughout time.

Also, in MCEC the influencing power of each player is restricted by its own subset of possible actions. As such, EC becomes a *collaborative* game in this multiplayer version—the players must cooperate, each contributing to different aspects of the city's health, so to achieve a sustainable city. Although the actions of a player implicitly impact each of the scores in a distinct manner, the objective of MCEC is the same as the single-player's version, *i.e.*, finish the game with enough population in a sustainable city. However, this does not mean that each player must maximize its associated score. In fact, despite being more "targeted" at improving some aspect of the game, most of the available actions influence several game scores and the amount of available resources when executed. They also critically impact the future states of the game since structures, upgrades and policies cannot be "undone" (*e.g.*, by building the "wrong" structure one may end up with no available space to increase the game's level). This fact makes the game inherently non-competitive, but makes it difficult to consider "good" plays for each of the game roles given *any* situation of the game.

Another challenge in the modeling of the robotic tutor's behavior is that he is able to play any of the roles in MCEC. The AI module has therefore to have the sufficient task knowledge that allows it to generate the *best game plays* given any possible situation of the game. This is also useful for the tutor's purposes as it allows him to play "as if he was one of the other player's" and thus develop theory-of-mind reasoning and propose action alternatives when suitable. There are also *real-time restrictions* associated with the AI module—it has to provide information on actions and other game-related information in a timely manner so to avoid "breaks" on the flow of interaction with the humans subjects [6].

D. Analysis of MCEC

We now analyze MCEC's topology as a game. As mentioned above, the *players' goal* is to progress to level 4 of the game having a population level of 200 and a sustainable city.[3] As such, the *utility function* for this game is one that provides a positive value to terminal states satisfying this condition, and a negative value to all other terminal states, *i.e.*, when $oil < 0$. Moreover, we can define MCEC as a *common interest* game, *i.e.*, all the players benefit if they cooperate in order to achieve the "common-goal". If we consider the game's *state* as the state of the city—it's current structures, upgrades, implemented policies and resources levels—, it is also a game of *perfect information* for the players as the environment is *fully-observable*, as depicted in Fig. 2, and all the actions are *deterministic*, *i.e.*, there is no stochasticity of any sort [7].

Before going into the mechanisms behind the AI Module responsible of controlling the autonomous game actions of the robotic tutor it is important to assess MCEC's game complexity. Determining such characteristic for MCEC analytically is a hard task given the complex inter-effects that each action has on the several game components and the rules associated with game termination. As such, we devised a Monte Carlo experimental procedure in which we performed a series of random-walks over the state-space of the game in the following way. We performed a total of 10^6 games of MCEC with a random play order between the roles. At each turn, we chose a random action for the respective player and the game ended whenever a terminal state was reached. We recorded the mean values for the *game length* in terms of number of actions performed during the game, and the number of available actions that each player/role has at each turn. This allows us to estimate the average *depth of the goal node* and the average *branching factor*. We can thus estimate the complexity of searching for solutions in MCEC given a representation of successive state transitions in a tree form [7]. The results of this study, during which approximately 23×10^6 states were explored, are reported in Table I. As we can see from the values in Table I, finding a solution in MCEC involves searching a game-tree with approximately $88.4^{23.8}$ (or 2^{46}) nodes.[4]

TABLE I. MEAN VALUES FOR THE GAME LENGTH (NUMBER OF TURNS UNTIL GAME TERMINATION) AND NUMBER OF ACTIONS *per* TURN AVAILABLE FOR EACH ROLE IN MCEC. RESULTS WERE TAKEN FROM 10^6 MONTE CARLO TRIALS. SEE TEXT FOR DETAILS.

Metric		Mean Value
Game length		23.8 ± 16.4
Actions	Economist	113.3 ± 186.8
	Environmentalist	68.7 ± 100.2
	Mayor	83.4 ± 125.7
	Player average	88.4 ± 143.2

III. COLLABORATIVE AI FOR MCEC

In this section we detail the proposed AI module for the MCEC game and its components.

A. AI Module Requirements

We recall that the main objective of the work reported in this paper was the construction of an AI module capable of controlling the game actions and informing the overall behavior of a robotic tutor playing the game of MCEC with two human subjects sharing the same physical space. We therefore created our AI module for MCEC revolving around four main objectives:

1) *Contribute* to a *sustainable* city by performing plays that best fit the game's current situation;
2) *Interact* with the human subjects in a timely manner, providing the required useful information to other control modules in *real-time*.
3) *Influence* the decisions of the human players by providing useful information about *attention-needing* aspects of the city, information on *why* a certain action was played or possible *alternatives* for actions made by the players;

[3]The experimental study conditions for the "robotic tutor" scenario described here state that the human subjects playing MCEC are explicitly informed of the "common goal" behind the game, *i.e.*, end-up with a sustainable city, and that the tutor is performing his actions accordingly.

[4]Notably, on average, a subject playing the role of Economist has a larger set of available actions in each turn, despite all players having the same amount of structures to build. This is due to the greater amount of updates available for economy-related structures, that only the Economist can perform.

4) *Adjust* the tutor's game strategy in a social manner throughout the game to be able to follow *group preferences*;

From the enumeration above we can derive a series of requirements that the AI module for MCEC must meet to provide significant results. The first and most obvious one is that the module must be able to provide the robot controller with game actions that contribute to the "common goal" given *any state of the game* (Obj. 1). Moreover, because the interaction scenario does not impose any specific game role to the robotic tutor, the AI module must be able to generate actions for *all roles*.

Another requirement is that of *real-time interaction* (Obj. 2), as discussed earlier—the robotic tutor must interact with the human subjects in a "natural" manner, playing the game "just like a human player would do", and also providing *useful* and *accurate* information about the city's status in terms of sustainability in a timely manner. Specifically, we defined a time limit of 5 seconds for the AI module to provide all the required information at each turn, including the turns in which the tutor is not performing a game action (*e.g.*, it may still provide suggestions about possible plays or alert for specific aspects of the city's health).

A major aspect of the robotic tutor's behavior within the interaction scenario being developed is that of *pedagogy*—the role of the tutor is not only to explain the game rules and be able to "play the game well", but also *pump discussion* about relevant aspects of the task, *e.g.*, raise the subject's environmental awareness about energy consumption, and create *interesting learning situations*, *e.g.*, make a "bad" game move on purpose to point something interesting about the effects of unsustainable structure building (Obj. 3). As such, the ability to *explain* the status of the city's health and each of the decisions made by the tutor is an important requirement of the AI module. The robot must also *adjust its game-play* decisions to best fit the "group strategy" being played by the human subjects during the game (Obj. 4), *e.g.*, detecting that the subjects and employing an "environment-aware" strategy focusing on actions that raise the environment score.

B. AI Module Architecture

The architecture for the AI Module proposed in this paper is depicted in Fig. 3, where we include the connection between the module and the other components involved. As we can see, a *game-playing component* is responsible for informing the *robotic tutor's decision-making mechanism* of predicted game values and game actions calculated according to the current game context. In turn, this controller uses the provided information to express interactive content in the form verbal, *e.g.*, say something related to the game to the human subjects, and non-verbal, *e.g.*, perform some animation while speaking, point in the direction of the game table, behaviors.[5] It also communicates with the MCEC game engine to perform game actions. From the game engine, the AI Module collects information on the several game values, *i.e.*, scores, resources, etc., and the players' game moves in order to keep its internal information on the city's status and player strategies up-to-date. The module comprises also a social component modeling the

[5]More details on this component can be found in [5].

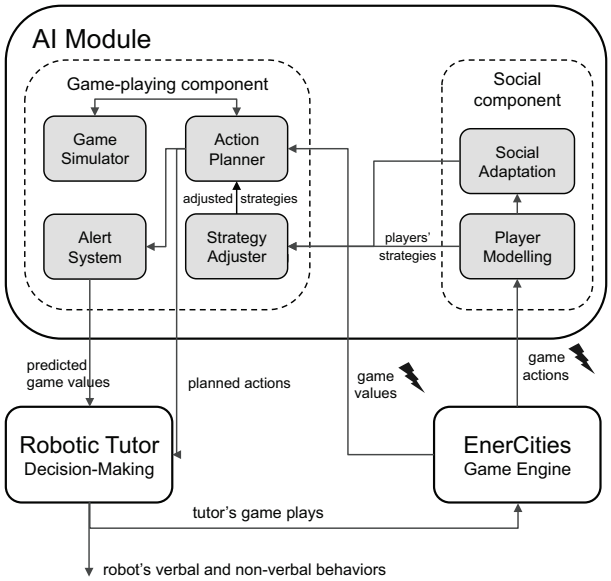

Fig. 3. The architecture of the AI module and overview of communication with the robotic tutor's decision making module and the MCEC game engine.

strategies being played by the human subjects, adjusting the tutor's strategy accordingly.

Before detailing each of the AI Module's components, we present a central concept of our approach that is used throughout our solution by almost all sub-components, that of *game strategy*. As explained earlier, in MCEC there is no numerical value translating a player's goal. Therefore it is hard to define a player's strategy in the game as one maximizing a given quantity. Instead, reaching the desired terminal state of the game, *i.e.*, a sustainable city, involves the interplay of multiple aspects that are influenced by each kind of action available. Furthermore, given the requirements associated with the robotic tutor in our scenario, one cannot strictly define a single strategy per existing role, as the robot's strategy generally depends on the game context and pedagogical factors. Another objective of the scenario is that the tutor is able to follow the strategy of the human players during the game so that the robotic agent is perceived as acting in an empathic and social manner [8]. All these factors make it hard to define a finite set of possible strategies beforehand.

Given such restrictions, in this paper we model a strategy in MCEC as a *tendency* to act upon the several aspects of the city. Formally, we define a strategy as a vector Θ given by:

$$\Theta = [\theta_v, \theta_e, \theta_w, \theta_m, \theta_p, \theta_o, \theta_h, \theta_u], \quad (1)$$

where θ_v, θ_e, θ_w, θ_m, θ_p, θ_o, θ_h and θ_u are respectively the *weights* associated with the *environment score*, *economy score*, *well-being score*, *money level*, *power level*, *oil level*, *population (homes) level* and *score uniformity*. Each weight component θ_i indicates the *importance* that the player's strategy Θ attributes to the corresponding game component. For example, a game strategy of $\Theta_{env} = [1, 0, 0, 0, 0, 0, 0, 0]$ means that a player is only concerned about the environment's score and will therefore chose actions that maximize the health of the environment. Also, a weight-vector $\Theta_{rnd} = [0, 0, 0, 0, 0, 0, 0, 0]$ represents *random strategy* in which none of the components is important to make decisions in the game. Game uniformity, as the name suggests, indicates the player's preference in having an equal

value in all the scores, or, in other words, a state in which the variance in the game's scores is low. A strategy can therefore be seen as a set of *preferences* for a player to deal with the several aspects related to the city's health. In our approach, we consider strategies such that $\theta_i \in [-1.0, 1.0]$ and $\|\Theta\|_1 = 1$, *i.e.*, vectors with a normalized absolute sum. In this manner, we define a continuous space for the game strategies, thus providing us the aforementioned flexibility in modeling player decisions in MCEC.

C. Game-playing Components

1) **Game Simulator:** The main purpose of this component is to manage the tutor's game-play in MCEC and inform the robot's controller of interesting aspects occurring within the game. As depicted in Fig. 3, this component includes three sub-components. As its name suggests, the *Game Simulator* is responsible to simulate the effect of executing all possible game actions given a certain state of the game. Therefore, we implemented a fully-functional game engine for MCEC according to the game rules mentioned earlier.[6] The simulator is mainly used by the action planner to generate successor states and thus simulate the value of all the game components. In addition, it is used to keep an up-to-date state of all structures, updates and policies currently implemented in the city according to the actions performed by the human players and the robotic tutor in the game.

2) **Action Planner:** The *Action Planner* is responsible of determining suitable game actions in order to achieve a desirable situation for the city. The planning procedure mechanism is a function of some strategy indicating a player's preference towards a future state of the game and the current values of all game components—henceforth referred to as the *game values*, as indicated in Fig. 3. The planner therefore maximizes the *gain* induced by the execution of an *action* given some *state* and a *strategy* being employed. Let us denote by $a_k \in \mathcal{A}_k$ an *action* performed by a player playing role k, where $\mathcal{A}_k$ is the space of all possible game moves for that role in MCEC. For the purposes of planning, we consider as *state* a vector $\mathbf{s} \in \mathcal{S}$ given by:

$$\mathbf{s} = [s_v, s_e, s_w, s_m, s_p, s_o, s_h, s_u], \tag{2}$$

where we used the same indexing notation as in (1) to refer to the several *game values*. $\mathcal{S}$ is the space of all possible states of the game values. As we can see, a state contains only information about game values, thus ignoring which particular structures are built and their position in the city, as well as the specific upgrades performed and policies implemented.[7] The value of the *score uniformity* component, s_u, is given by the *negative coefficient of variation* between the game's scores:

$$s_u = -\frac{Avg_k(s_k)}{StdDev_k(s_k)}, s_k \in \{s_e, s_v, s_w\}.$$

As stated earlier, the planner can generate suitable actions for any given role in any game state according to some strategy.

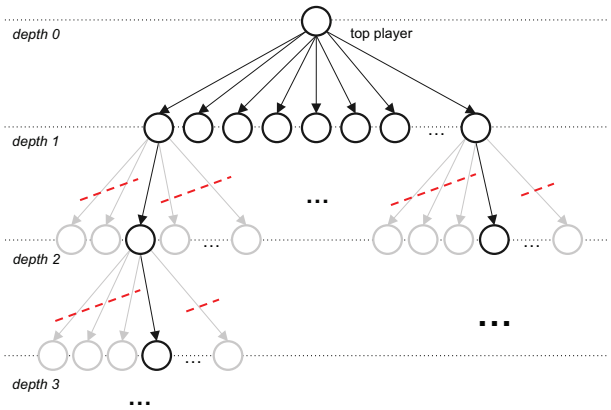

Fig. 4. Depiction of the tree-search procedure within the action planner. Nodes represent *game value states*, arrows represent state transitions caused by the execution of some *action*. Bold nodes and arrows represent maximal *gain* state transitions, while gray nodes and arrows slashed by a red line represent *forward prunning* performed during planning. See text for details.

In terms of the planning procedure, and given the analysis in Sec. II-D on the complexity of searching for optimal solutions in MCEC, we opted to perform a *heuristic-based* search. Let $T(\mathbf{s}, a) \rightarrow \mathcal{S}$ be the deterministic *transition* or successor function providing the values of the game state resulting of executing action a given state $\mathbf{s}$. The *gain* of executing a in $\mathbf{s}$ given strategy Θ is then a scalar given by:

$$\gamma(\mathbf{s}_t, \Theta, a) = Norm(T(\mathbf{s}, a) - \mathbf{s})^T \Theta, \tag{3}$$

where $Norm(\mathbf{s}) \rightarrow [-1, 1]$[7] is a function normalizing each component of a game values state, $s_i \in \mathbf{s}$, according to:

$$s_i = \frac{s_i - Avg(s_i)}{StdDev(s_i) + (Max(s_i) - Min(s_i))/4}, \tag{4}$$

where the statistics $Avg(s_i)$, $StdDev(s_i)$, $Max(s_i)$ and $Min(s_i)$ are calculated using *all* values simulated for component s_i during the game session.[8]

The resulting planning procedure is similar to that of *MIN-MAX* [7] in a non-adversarial game, with *forward pruning* of nodes at each planning step. The planner searches for the optimal game action, denoted by a_k^*, for a player playing role $k \in \{eco, env, may\}$. It takes as input the current game values state, $\mathbf{s}$, a set of strategies for all players, $\mathbf{\Theta} = \{\Theta_{eco}, \Theta_{env}, \Theta_{may}\}$, the *maximum depth* that planning can reach, denoted by δ_{max}, and the *maximum time* available to perform planning, denoted by τ_{max}. Let $NextRole(k)$ be a function returning the player role following k according to some predefined playing order, and $TimePassed(\tau)$ be a function returning the time passed since time instant τ.

The planning algorithm proposed for MCEC is presented in Algorithm 1 and a tree-form depiction of the planning procedure is presented in Fig. 4. We refer to *top player* the player for which we are calculating the optimal game move, a_k^*. As we can see, the procedure starts by getting the *1-depth look-ahead gain* for each of the top player's available actions (steps 2-4).[9] The gains are then sorted in descending order—

[6]We opted to implement the simulation engine from scratch as the original game engine for MCEC was tightly implemented within its graphical engine, preventing its practical use for planning purposes.

[7]We recall that the terminal conditions for MCEC only refer to values of the game components, *i.e.*, scores, population and oil level, independently of the particular actions performed to reach such state.

[8]This function captures the *quality* of each game value in terms of being positive or negative compared to what the *expected value* for that component is. We take advantage of simulated values to calculate the statistics for each game value component.

[9]If a certain action is not executable in some state, *e.g.*, building a level-3 structure while in game level 1, that action is discarded from the set.

Algorithm 1 Planning procedure for player k

Input: state $\mathbf{s}$, strat. set $\mathbf{\Theta}$, max. depth δ_{max}, max. time τ_{max}
Output: $a_k \in \mathcal{A}_k$

1: $\tau = 0$
2: **for** a_k in $\mathcal{A}_k$ **do**
3: $\quad \Gamma(a_k) = \gamma(\mathbf{s}, \Theta_k, a_k)$ ▷ Get gains for top player actions
4: **end for**
5: $SortDesc(\Gamma)$ ▷ Sort action gains in descending order
6: $a_k^* = \emptyset$, $\gamma^* = -\infty$
7: **for** a_k in $\mathcal{A}_k$ **do**
8: $\quad j = k$, $\mathbf{s}_\tau = \mathbf{s}$, $\delta = 0$ ▷ Reset variables
9: $\quad$ **while** $\delta < \delta_{max} \wedge \tau < \tau_{max}$ **do**
10: $\quad\quad j = NextRole(j)$
11: $\quad\quad$ **for** a_j in $\mathcal{A}_j$ **do**
12: $\quad\quad\quad \Gamma_j(a_j) = \gamma(\mathbf{s}_\tau, \Theta_j, a_j), \Theta_j \in \mathbf{\Theta}$
13: $\quad\quad$ **end for**
14: $\quad\quad a_j^* = \text{argmax}_{a_j} \Gamma_j(a_j)$ ▷ Update state
15: $\quad\quad \mathbf{s}_\tau = T(\mathbf{s}_\tau, a_j^*)$
16: $\quad\quad \gamma_\tau = Norm(\mathbf{s}_\tau - \mathbf{s})^T \Theta_k$ ▷ Get top player gain
17: $\quad\quad$ **if** $\gamma_\tau > \gamma^*$ **then** ▷ Verify max gain
18: $\quad\quad\quad \gamma^* = \gamma_\tau$
19: $\quad\quad\quad a_k^* = a_k$
20: $\quad\quad$ **end if**
21: $\quad\quad \delta = \delta + 1$ ▷ Update depth and time
22: $\quad\quad \tau = \tau + TimePassed(\tau)$
23: $\quad$ **end while**
24: **end for**
25: Return a_k^*

the idea is that if planning time runs out, we expand what seem to be most promising branches in the search-tree first. Then, for each top player action, a_k, we expand search by simulating game-playing turns until the maximum number of turns, δ_{max}, is reached, or we run out of time to plan (steps 7-24). In each turn, we calculate the *1-depth look-ahead gains* of all possible actions a_j of player j, according to the given strategy Θ_j (steps 11-13). The procedure then performs *forward pruning* in the search tree by expanding only the branch of the action with the highest gain, a_j^*, in the perspective of the turn's player (steps 14-15). We also calculate the *τ-depth look-ahead gain*, denoted by γ_τ, in the perspective of the top player (step 16). The procedure then verifies whether such gain is more advantageous, in which case a_k^* is updated (steps 17-20). When planning is finished, the planner returns the action providing the overall maximum expected gain, a_k^*.

3) **Strategy Adjuster:** This sub-component is responsible for adjusting online the strategy of the player for which an action is going to be planned, as depicted in Fig. 3. It functions like a *hormonal-like* mechanism inspired in the way *neurohormones* modulate behavioral outputs in biological systems according to changes perceived in the environment [9]. With that in mind, we defined a series of *adjustment functions*, denoted by $\sigma_i(s_i, \varsigma_i)$, each modulating some component $\theta_i \in \Theta$ of a player's strategy Θ. For example, if the power level of the city is perceived as being low, the respective adjustment function will raise the power component of a player's strategy so that the planning procedure chooses actions leading to states with a higher power level. Each function receives as input the

TABLE II. ADJUSTMENT FUNCTIONS FOR STRATEGIES IN MCEC.

Component	Adjustment Function
Oil	$\sigma_o(s_o, \varsigma_o) = {\varsigma_o}^{s_o}$
Power	$\sigma_p(s_p, \varsigma_p) = {\varsigma_p}^{s_p}$
Money	$\sigma_m(s_m, \varsigma_m) = {\varsigma_m}^{s_m}$
Scores	$\sigma_k(s_k, \varsigma_s) = 0.2{\varsigma_s}^{s_k}, s_k \in \{s_e, s_v, s_w\}$
Population	$\sigma_h(s_h, \varsigma_h) = 1 - {\varsigma_h}^{s_h/200}$
Environment	$\sigma_e(s_v, \varsigma_e) = 0.2{\varsigma_e}^{s_v}$

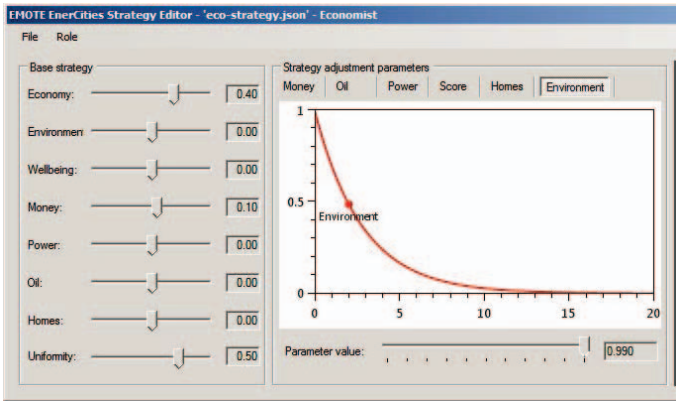

Fig. 5. The tool used to refine the strategy adjustment parameters.

respective game value, θ_i, while ς_i is an *adjustment parameter* influencing function's σ_i output. The set of all adjustment parameters is fixed during the game and predefined for each player, indicating his *predisposition* in dealing with attention-focusing events from the game. This component thus functions like a self-regulatory mechanism dynamically adjusting the player's strategy at each turn. We present the proposed functions used in our studies in Table II. The "Scores" function, σ_s^k, adjusts the respective score component related to each role k, *e.g.*, for the Mayor player it adjusts the well-being component, θ_w. The presence of function σ_e means that all players get their strategies' environment component, θ_v, adjusted—this is due to the fact almost all build actions in MCEC negatively impact the environment's score, and therefore this is a component requiring a higher degree of attention for any player role. After being adjusted, a strategy Θ is normalized to ensure that $\|\Theta\|_1 = 1$. We created a tool that helps refining the adjustment parameters and observe the modulation effect. A screenshot of the tool is presented in Fig 5.

4) **Alert System:** The purpose of this component is to inform the robotic tutor's decision-making of possibly interesting game situations so that he may intervene during the interaction with the human subjects, as indicated in Fig. 3. One of such information provided by the alert system is the *predicted game values*, calculated by averaging all simulated $\mathbf{s}_\tau$ during the planning procedure described in the section above. The idea is to provide a "glimpse" of what the game values may be in the game's near future and act upon them when certain thresholds for each component are reached—for example, if the probability of reaching a low energy level is high, the robot may refer to "energy saving strategies" or suggest the construction of an energy-related structure. The component also provides information about the *no-action*

probability, *i.e.*, the probability of reaching a state where players have no option other than *skip turn* in the near future. This probability is again calculated using information collected during planning, by averaging the number of times players only had a skip-turn action in some simulated turn. Finally, the system also alerts about the *no-space probability* in the game, *i.e.*, the probability of reaching a state where *no player* has an available cell on which to build a new structure. This value is calculated according to the ration between the current number of empty cells in the city and the standard number of available cells in each level. These two probabilities may be used by the tutor *e.g.*, to inform the human subjects of the urgency in building residential structures in the city in order to pass the game level and thus obtain more available space.

D. Social Component

The main purpose of this component is to understand the game-related behavior of the human subjects playing MCEC with the robotic tutor.

1) **Player Modeling:** This sub-component is responsible for modeling the game strategies of the human players. To achieve that, and for reasons of simplicity, we consider that each human subject is trying to maximize the *1-step look-ahead gain* when playing a specific action in some game turn, *i.e.*, we assume that players are trying to maximize the *immediate* gains of their actions. As depicted in Fig. 3, whenever some player k makes a game action, a_k, this sub-component registers the normalized difference, denoted by $\Delta\mathbf{s}$, in the game values resulting from such state transition, *i.e.*,

$$\Delta\mathbf{s} = Norm(T(\mathbf{s}, a_k) - \mathbf{s}),$$

where each game values component is normalized according to (4). Then, it updates its current estimate of player k's strategy, Θ_k, according to:

$$\Theta_k = \Theta_k + \alpha.\Delta\mathbf{s}, \tag{5}$$

where α is a *learning rate*, ensuring a smooth variation after each play. Each player's strategy is learned online, *i.e.*, while the human subjects are playing a game session. They are used by the game planner to predict the decisions of the human subjects during planning, resulting in a *theory-of-mind* (TOM) reasoning performed according to observed data from the several game plays. Initially, each strategy is set according to some "expected" behavior according to the player's role, *e.g.*, one may expect that a player playing the role of Economist to value more the economy score and the money level.

2) **Social Adaptation:** Another purpose of the social component is to adapt the game-play strategy of the robotic tutor in a social manner, *i.e.*, according to the strategies estimated for the human players, thereby adhering to the perceived *group strategy*. Because the game of MCEC is inherently collaborative, we also assume that the human players are adjusting their strategies to one another.[10] Therefore, after a human player performs some game action, this sub-component "approximates" the other human player's estimated strategy and the tutor's strategy according to:

$$\Theta_k = \Theta_k + \lambda(\Theta_j - \Theta_k), \tag{6}$$

where Θ_k is the player whose strategy is being adjusted, Θ_j is the player that performed the game action, and λ is an *imitation factor* regulating the strategy approximation.

[10]By observing the interactions between the human subjects during the game sessions, we verified that indeed they often discuss their game strategies and ways to collaborate in the game, *e.g.*, by "following" a money-saving strategy.

IV. Discussion and Conclusions

A. Application within EMOTE

In this paper we presented an AI module for MCEC, a multiplayer collaborative game about environmental awareness, in the context of project EMOTE, in which an emphatic robotic tutor interacts with human players in a social and pedagogical manner. Within this context, we created an AI module that endows the tutor of not only *explaining* his own actions but also *raising discussion* on important topics related with the game. We have been performing studies with human subjects interacting and playing with our robotic tutor in MCEC and we have received a very positive feedback.

To achieve such rich behavior on the robot's part, our proposed AI module includes two main components with distinct functionality. The game-playing component is able to *simulate* and keep *updated information* about the game, *e.g.*, on the resources, game scores, etc. It is capable of *planning* actions for any game role according to predefined criteria, *i.e.*, the *game strategy*. In the context of EMOTE, this enables us to *personalize* the robotic tutor's behavior according to some pedagogical objective, *e.g.*, we may define a condition in which the tutor plays as "energy saver", raising awareness on building low energy-consumption structures, and another condition in which he plays a "spare-no-expenses" strategy to create interesting pedagogical situations. Also, the *alert mechanism* has allowed us to define thresholds for some components, that when reached give the tutor the opportunity to focus the human subjects' attention into solving the problem at hand. Moreover, this AI component is being used to *justify* the tutor's choices in the game, *e.g.*, by looking at the components of the strategy used, comparing the alternative actions resulting from planning and explaining why the best was chosen. Overall, our action planner supports the conjunction of predefined profiles or predispositions—*static* elements, with online imitation and a hormonal-like, attention focusing system—both *dynamic* components. The preliminary results of our studies suggest that indeed this component allows the robotic tutor to be perceived as more "intelligent" and "alive" by the human subjects, by being *aware* of events occurring within the game and responding accordingly.

Our proposed AI module also includes a social component that is able to perceive and inform on the *intentions* and strategies of the human players. This endows our robotic tutor of a *TOM component* that uses the predicted strategies for the planning of the human subjects' simulated game-play turns. In the context of our studies in EMOTE, such social component has allowed the tutor of commenting on players' choices and proposing *alternative actions*, *e.g.*, by planning in the perspective of a human subject according to his estimated strategy, and then comparing the planned action against the one that was executed by the human player. This component

TABLE III. RESULTS OF THE "SELF-PLAY" EXPERIMENTS.

Group	**Data Collected**									
Strategy	l	δ	s_h	s_o	s_m	s_p	s_v	s_e	s_w	u
Balanced	4	33	209	14.6	95	6.5	3.5	8.5	11.0	4
Life-qual.	4	36	203	117.0	122	7.5	2.5	12.5	20.5	7
Score-grd.	3	45	124	−67.0	104	0.5	50.5	11.0	33.0	7
Spender	3	27	142	−30.0	98	33.0	−26.5	22.5	7.5	0

includes also a social imitation mechanism that makes the tutor follow the perceived *group action tendency*. This makes our robotic tutor to be perceived as more "socially aware", even when not performing "optimally" in the game. *I.e.*, it is preferable for the tutor to follow the common group strategy than to always perform the optimal action leading the game faster to its termination. According to the feedback given by the human subjects, this has to do with the tutor being perceived as more *social* and making *less unexpected* actions that deviate from the group strategy being followed.

B. Empirical Study

Apart from the studies performed with human subjects in the context of EMOTE, we also performed more empirical studies on the proposed AI module. The objective was to test its planning, personalization and strategy imitation capabilities. We devised a *self-playing* tool in which the AI module autonomously controlled the actions of all three players. We then run several game sessions using this tool and set distinct strategy profiles for each group of players. Specifically, we designed four different profiles: a *balanced* strategy, as its name suggests, tries to balance all game values; a *life-quality* strategy results in a preference for environmentally-aware actions; a *score-greedy* strategy makes each player to try gather as much of its own score as possible and a *spender* strategy is directed at advancing in the game's levels as quickly as possible disregarding the cost associated with the actions. We note that in the case of the spender test, the players strategies were taken from the values estimated by the player modeling sub-component, during a game session with human subjects deliberately instructed to follow a "spending" behavior. The purpose of this experiment was to test the action planner ability in developing different kinds of cities, resulting from the actions played by groups of players playing according to distinct strategy profiles. The results of this experimental study can be found in Table III, where we present the data collected at the end of the simulations.[11] We present all components of the game values except s_u. l is the level reached in the game, δ is the number of game turns played and u is the total number of upgrade actions performed.

As we can see, both the balanced and life-quality conditions lead to sustainable cities at the end of the simulations, the main difference being that the life-quality strategies achieved higher game values in general and more upgrades performed. By looking at the structures built in each test we are able to observe a much higher concern in the life-quality condition in building environment-friendly structures and green spaces that end up saving natural resources. In the other two tests, the strategies employed lead to a generalized poor performance in terms of the city's health. The score-greedy condition obtained, as expected, the overall higher scores. The final structure shows us a very diverse city in which each player tried to built as much of their role-related structures as possible. However, this made the players somehow ignore the game level progressing by having too few residential structures built. Finally, the "imitated" strategies in the spender condition obtained the lowest number of turns in the game. This was due to the construction of expensive and resource-wasting structures thus disregarding oil consumption. We can also see the effect of not performing upgrades in the game, which makes the structures sub-optimal in terms of resource consumption.

[11] Videos of the game sessions as well as the specific strategies and adjustment parameters used can be found at: `http://gaips.inesc-id.pt/~psequeira/videos/enercities-ai/`.

C. Future Work

In the future, we are going to expand the AI module presented here to be able to fully control the robotic tutor's behavior while interacting with the human players during the game. We intend to use the information collected during our Wizard-of-Oz studies, in which a human expert controls all the behavior (verbal and non-verbal) of the robot, and create machine learning mechanisms within the AI module that are able to "capture" the expert's decision-making process. In turn, a behavior selection mechanism will select the appropriate behavior whenever some situation is detected, either related with the MCEC game itself or with the social interactions occurring externally to the game.

ACKNOWLEDGMENT

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013 and by the EU-FP7 project EMOTE under grant agreement no. 317923. The first author acknowledges a BDP grant CMUP-ERI/HCI/0051/2013 from project INSIDE, funded by the FCT.

REFERENCES

[1] G. Castellano, A. Paiva, A. Kappas, R. Aylett, H. Hastie, W. Barendregt, F. Nabais, and S. Bull, "Towards empathic virtual and robotic tutors," in *Artificial Intelligence in Education*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, vol. 7926, pp. 733–736.

[2] E. Knol and P. De Vries, "EnerCities: Educational Game about Energy," in *Proceedings of Central Europe towards Sustainable Building 2010*, ser. CESB 2010, Prague, Czech Republic, 2010, pp. 12–15.

[3] A. Brisson, G. Pereira, R. Prada, A. Paiva, S. Louchart, N. Suttie, T. Lim, R. Lopes, R. Bidarra, F. Bellotti, M. Kravcik, and M. Oliveira, "Artificial intelligence and personalization opportunities for serious games," in *8th Art. Intel. and Interactive Dig. Entert. Conf.*, ser. AIIDE 2012. Palo Alto, CA, USA: AAAI Press, October 2012, pp. 51–57.

[4] G. W. Brown, "Some notes on computation of games solutions," RAND Corporation, Santa Monica, CA, USA, Research Memoranda RM-125-PR, 1949.

[5] T. Ribeiro, A. Pereira, A. Deshmukh, R. Aylett, and A. Paiva, "I'm the mayor: A robot tutor in enercities-2 (extended abstract)," in *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, ser. AAMAS '14, 2014, pp. 1675–1676.

[6] C. Sidner and M. Dzikovska, "A first experiment in engagement for human-robot interaction in hosting activities," in *Adv. Natural Multimodal Dialogue Systems*. Springer, 2005, vol. 30, pp. 55–76.

[7] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. Prentice Hall, 2003.

[8] EMOTE project: EMbOdied-perceptive Tutors for Empathy-based learning. [Online]. Available: http://www.emote-project.eu/

[9] R. Harris-Warrick, E. Marder, A. I. Selverston, and M. Moulins, Eds., *Dynamic Biological Networks*. The MIT Press, 1992.