Real-Time Strategy Experience Exchanger Model [Real-See]

Mostafa Aref [1],Magdy Zakaria [2]and Shahenda Sarhan [3]

[1] Faculty of Computers and Information, Ain-Shams University
Ain-Shams,Cairo,Egypt

[2] Faculty of Computers and Information, Mansoura University
Mansoura,Egypt

[3] Faculty of Computers and Information, Mansoura University
Mansoura,Egypt

Abstract

For many years, researchers tried and succeeded to develop agents that can adapt their behavior to face new opponent scenarios and beating them. So in this paper we introduce an experience exchanging model that allow a game engine to update all other engines with the game reaction against new surprising un-programmed opponent scenarios that face the computer player through exchanging new cases among engines case-based reasoning systems. We believe this will reveal game players from downloading a new engine of the game and loosing their saved episodes.

Keywords: *Real-Time Strategy Games, Case-based Reasoning, Feature Similarity.*

1. Introduction

Artificial Intelligence (AI) [2][4][18] is the area of computer science focusing on creating intelligent machines. The ability to create intelligent machines has intrigued humans since ancient times. Today with the advent of the computer and 60 years of research into AI programming techniques, the dream of smart machines is becoming a reality.

Researchers are creating systems as intelligent agents that can autonomously decide about the desired results without user interaction, script or even fixed execution plan. They can mimic human thought, understand speech and beat the best human chess-player. This has two benefits, first, they allow for a high-level definition of the problem. Secondly, agents are better reusable and more robust than fixed programs. These benefits make agents a suitable area for computer AI games.

AI games has existed since1951 when Christopher Strachey wrote a checkers program [16][18]. As 3D rendering [16] hardware and resolution quality of game graphics improved, AI games had increasingly become one of the critical factors determining a game's success. From this we can refer to AI games as techniques used in computer and video games to produce the illusion of intelligence [16][18] in the behavior of non-player characters (NPCs). While the non-player character is a character that is controlled by the game master so it is a part of the program, but not controlled by a human.

The real-time performance requirements of computer AI games, the demand for humanlike interactions [5], appropriate animation sequences, and internal state simulations for populations of scripted agents have impressively demonstrated the potential of academic AI research and AI games technologies.

2.Background

2.1 Real-Time Strategy Games

A real-time strategy game (RTS) is a strategic war [5][9] game in which multiple players operate on a virtual battlefield, controlling bases and armies of military units. It typically ends with the destruction of the enemy.

The better balance you get among economy, technology, and army, the more chances you have to win.

Although many studies exist on learning to win games with comparatively small search spaces, few studies exist on learning to win complex strategy games. Some researchers argued that agents require sophisticated representations and reasoning abilities to perform well in these environments, so they are challenging to construct.

Fortunately, Ponsen and Spronck (2004) [14] developed a good representation for WARGUS, a moderately complex RTS game. They also employed a high-level language for game agent actions to reduce the decision space. Together, these constrain the search space of useful plans and state-specific sub-plans, allowing them to focus on the performance task of winning RTS games.

Marthi, Russell, and Latham (2005) [11] applied hierarchical reinforcement learning (RL) in a limited RTS domain. This approach used reinforcement learning augmented with prior knowledge about the high-level structure of behavior, constraining the possibilities of the learning agent and thus greatly reducing the search space.

Ponsen, Muñoz-Avila, Spronck and Aha (2006) [12] introduced the Evolutionary State-based Tactics Generator (ESTG), which focuses on the highly complex learning task of winning complete RTS games and not only specific restrained scenarios.

2.2 Case-based Reasoning

Case-based Reasoning (CBR) is a plausible generic model of an intelligence and cognitive science-based method by the fact that it is a method for solving problems by making use of previous, similar situations and reusing information and knowledge about such situations. CBR [13] combines a cognitive model describing how people use and reason from past experience with a technology for finding and presenting such experience. The processes involved in CBR can be represented by a schematic cycle as shown in figure (1).

1. **Retrieval** is the process of finding the cases in the case-base that most closely match the current information known (new case) [1][8].
2. **Reuse** is the step where [1] matching cases are compared to the new case to form a suggested solution.
3. **Revision** is the testing of the suggested [8] solution to make sure it is suitable and accurate.
4. **Retention** is the storage of new cases for future reuse.

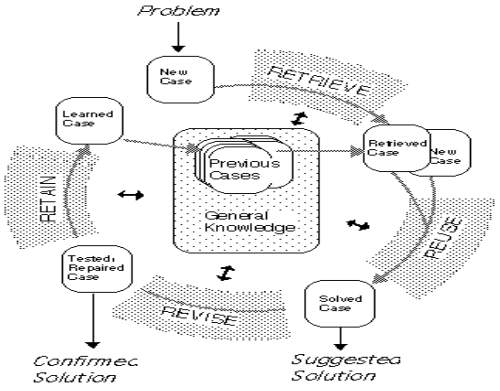


Fig.1 Aamodt Case-based reasoning cycle [1]

2.2.1 Case-based Reasoning related to RTS

In this section we will try to summarize some case-based reasoning researches on real-time and/or strategy games. Some CBR researches has targeted real-time individual games, as Goodman's (1994) [7] projective visualization for selecting combat actions, and predicting the next action of a human playing Space Invaders.

MAYOR (1996) [6] used a causal model to learn how to reduce the frequency of failed plan executions in SimCity, a real-time city management game. Where Ulam et al.'s (2004) [17] meta-cognitive approach performs failure-driven plan adaptation for Freeciv game. They employed substantial domain knowledge, and addressed a gaming sub-task (i.e., defend a city).

Molineaux and Ponsen (2005) [2] relax the assumption of a fixed adversary, and develop a case-based approach that learns to select which tactic to use at each state. They implemented this approach in the Case-based Tactician (CAT). They reported learning curves that demonstrate its performance quickly improves with training, even though the adversary is randomly chosen for each WARGUS game. CAT is the first case-based system designed to win against random opponents in a RTS game.

Santiago et.al.,(2007) proposed Darmok [15] as the base reasoning system, which is a case-based planning system designed to play real-time strategy (RTS) games. In order to play WARGUS, Darmok learns plans from expert demonstrations, and then uses case-based planning to play the game reusing the learnt plans.

In this section, different concepts and topics related to RTS games were explained. All challenges that face RTS games were concerned with increasing game intelligence through improving tactics, reinforcement learning, player satisfaction and modeling opponents. But our concern

was different; we tried to increase game intelligence not through learning but through exchanging experiences between game engines. That we will try to explain in next section.

3. Real-Time Strategy Experience Exchanger Model [Real-See]

As usual if you want to update any application you just need to download its update from its web site but what would you do if your engine of the application is more updated than the source itself ?!.Usually this cannot happen in ordinary applications, but here we are talking about RTS games which depend on agents trained by the recent RL techniques. This means that they can update themselves according to any changes in their environment.

In this paper we introduce our model that allowed an RTS game engine to update all other engines with the game reaction against new surprising un-programmed opponent scenarios that face the computer player. We believe this will reveal game players from downloading a new engine of the game and loosing their saved episodes. But we first needed to discuss the existing case representations and whether we can use them or we will need one of our own.

3.1 Proposed Case Representation

Many case representations are depending on the game or the researcher point of view. We here tried to make use of the former representations to get a case representation that suits our model and could be applied in different RTS games. For example Aha et.al (2005) [2] defined a case C as a four-tuple:

C = [BuildingState, Description, Tactic, Performance]

Where we can consider the BuildingState as a part of the Description. We can also notice that they didn't mention the goal of the case while it is an important factor in case retrieval. From all of this we proposed a case representation of our own to use it through our model

C =<State, Action, Goal, Problems to avoid, Performance >

- State is a vector composed of features representing game state that the system has already experienced.
- Action set is a list of case actions the agent can take at that level in the architecture.
- Goal: is a list of Goals to be achieved
- Problems to avoid: is a list of Problems to avoid
- Performance is a value in [0, 1], reflects the utility of choosing that tactic for that state.

Our case representation concentrates on making case retrieval more accurate and easier depending first on the case state features then on goal and performance. We here used the famous Missionaries and Cannibals problem as an example of our proposed case representation as following:

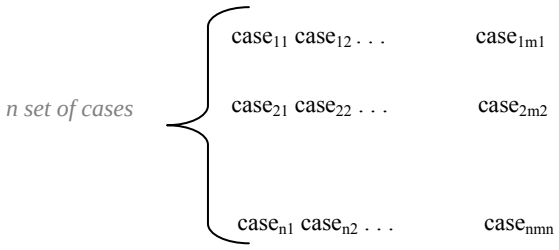
- **State = <M, C, B, P>**
  State = <3, 3, 1, 2>

  Where M: no. of missionaries
  C: no. of Cannibals
  B: no. of boats
  P: no. of people a boat can accommodate at a time
- **Actions**

  Move (D_1, D_2)
  Return (D_1, 0)
  Move (S_1, S_2)
  Return (S_1, D_1)
  Move (S_1. S_3)
  Return (D_2,0)
  Move (D_2, D_1)
  Return (D_2, 0)
  Move (D_2, D_3)
- **Goals**: Cross the river
- **Problems to avoid** : Cannibals eat Missionaries
- **Performance**: Less time to solve the problem equals higher performance.

3.2 Real-See Model

We supposed that n sets of cases from N engines were sent to the receiver engine figure (2). Each set consists of M_n cases.



These cases represent the input of the case comparator. The case comparator compare each case of them with the cases in the case-base that most closely match the current information known, and if it found a match it discards the received case and repeat the operation on the next

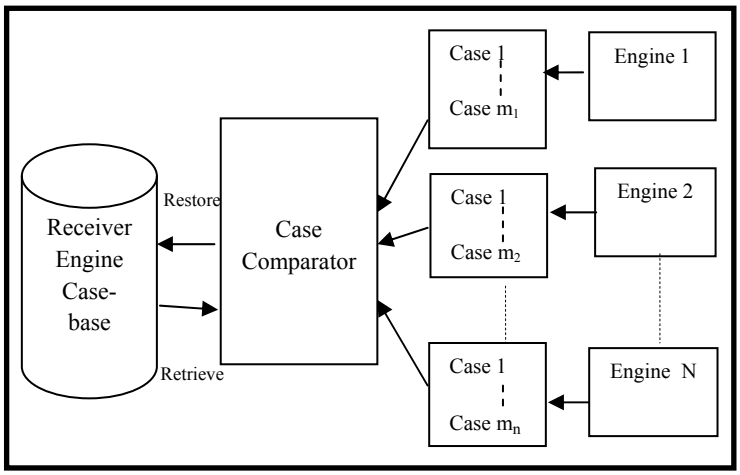


Fig.2 **Real**-See Model

one till it finishes all the M*N cases. The cases that didn't have a match in the case-base will be stored in the receiver engine case-base and the rest will be deleted.

In Real-See model the case comparator plays the major role as it dose all the job. In the next section we will discuss the case comparator in details.

3.2.1 Case Comparator

The case comparator compare each received case with the cases in the case-base, in order to do that we will need to make use of the similarity metrics. If the case comparator did not found a similar case to the received one it will add it to the case-base but if it found a similar one it will act according to the similarity degree.

Given a received case P, the matching of case P and a retrieved case C is guided by the similarity metric in equation (1).

$$\text{similarity}(P,C) = \frac{\sum_{i=1}^{k} w_i \times \text{sim}((p_i, c_i))}{\sum_{i=1}^{k} w_i} \quad (1)$$

Where w_i is the weight of a feature *i*, sim is the similarity function of features, and p_i and c_i are the values for feature *i* in the target and retrieved cases respectively.

But before calculating cases P and C similarity, we first needed to calculate the value of individual features similarity, sim(p_i,c_i). The feature *i* similarity of both cases P and C is related to the distance between them. Many equations were used to calculate the feature similarity depending on the distance, for example

- Euclidian distance [10][18]

$$d(P,C)= \sum_{i=1}^{k} \sqrt{p_i^2 - c_i^2} \quad (2)$$

- Hamming distance [10][18]

$$H(P,C) = k - \sum_{(i=1,k)} p_i \bullet c_i - \sum_{(i=1,k)} (1-p_i)\bullet(1-c_i) \quad (3)$$

- Absolute distance [18]

$$d(P,C)= \sum_{i=1}^{k} |p_i - c_i| \quad (4)$$

Here we chose to use the absolute distance divided by the feature values range specially that we are dealing with un-scaled discrete values not vectors, which is computed by:

- Distance for *Numeric* features

$$d_i(P,C) = |p_i - c_i|/(p_i + c_i) \quad (5)$$

- Distance for *Symbolic* features

$$d_i(P,C)= 0 \text{ if } p_i = c_i \quad (6)$$
$$= 1 \text{ otherwise}$$

From equations (5) and (6) we can say that

$$\text{Sim}(p_i,c_i)= 1- d_i \quad \text{where } 0 \le \text{Sim}(p_i,c_i) \le 1 \quad (7)$$

The next step is to calculate feature *i* weight. The feature weight may be calculated using many ways for example the distance inverse but this way will be a problem if the feature values were equal which means that the distance will be zero. Here we used the inverse of the squared standard deviation; as the standard deviation represents a sample of the whole feature values population and is a measure of how widely values are dispersed from the average value. In this case of feature values equality the weight is discarded and the feature similarity value will equal 1. We here calculated the weight using equation (8).

$$w_i= 1/\sigma(i)^2 \quad (8)$$

The last step is to calculate case P and case C similarity using equation (1), and to check its value relating to a threshold value α according to our Real-See algorithm in figure (3).

In figure (3), a received case P is retained as long as its similarity value relative to case C is not above α. As the result we get a set Q of retained cases as:

$$Q = \{P \in M_n \mid \text{Sim}(P,C) \le \alpha\}$$

Where M_n is the received cases and Sim (P, C) denotes the degree of similarity of C respect to P. The elements in Q along with their similarity scores are delivered to the receiver engine case-base for to be retained.

But what happened to the cases its similarity value relative to C is above α? Shall we decline them or what? Here in our model we tried to make use of the case goal.

Real-See Algorithm

For i=1 to n

Do for j=1 to M_n

$$similarity(Cij, C) = \frac{\sum_{x=1}^{y} w_x \times sim((Cij)_x, C_x)}{\sum_{x=1}^{y} w_x}$$

If similarity $(C_{ij},C) \leq \alpha$ Then case stored

Else

If $G(C_{ij}) \; !\Leftrightarrow G(C)$ Then case stored

Else

If $G(C_{ij}) \Leftrightarrow G(C)$ && $P(C_{ij})>P(C)$ Then case stored

Else C_{ij} to be discarded

Endif

Endif

Endif

Fig.3 **Real**-See algorithm

Till now similarity metrics depends on the case description. In our model this means to decline cases similar to the retrieved ones. So we tried to apply the similarity metrics on the case goals, if case P similarity value relative to case C is above α (α=0.5) the case comparator will compare case P and case C goals

But to calculate the goal similarity we first need to check the similarity of its parts. If there is a similarity we can express it by one else by zero. The calculated similarities is then applied in equation (9)

$$MoS = \frac{1}{R}\sum_{i=1}^{R} B_i \qquad (9)$$

Where B_i represents the predicate *i* of the goal, R is the number of predicates used in similarity calculation and MoS represents the arithmetic mean of the predicates similarities and we used it as the goal similarity. We can then evaluate the mean of similarities (Mos) using equation (10)

$$\text{Goal similarity} = \begin{cases} \text{Not Similar} & MoS \leq ½ \\ \text{Similar} & MoS > ½ \end{cases} \qquad (10)$$

If it found a goal match and case P performance is greater than case C performance, case P will be stored otherwise case P is declined. But if there was no goal match case P will be stored. We will explain it clearly in the next section with real picked cases.

4. Testing Real-See Model on Real Cases

Fig.4 Glest – 3D RTS game

For more explanation we needed to test the Real-See algorithm on some real cases. We selected a 3D RTS game called Glest figure (4) to pick up some cases of it to go one with our algorithm testing.

- **Example 1**: We first chose a stored case called the three towers (case C) to compare it with a received case called defend the castle (case P). In the next five steps we calculated the similarity between the two cases using 14 features (table 1) to representing each case.

  - The first step is to calculate feature *i* similarity. So we calculated the absolute distance using equations (5) and (7).
  - The second step is to calculate feature *i* weight using equation (8).
  - The third step is to calculate the similarity between case P and C using equation (1).

We can notice from table (2) that the features of value zero in both cases are discarded and were not contributed in the calculation, as it has no effect on the similarty degree which can finally be calculated as following:

Similarity (P,C)= 3.066/6.732=0.456

Table 1: The data set of Three_Towers and Destroy_Villag cases erepresenting14 features.

Features		*Three_Towers (Case C)*	*Defend the Castle (Case P)*
Resources	Gold	200	500
	Wood	200	500
	Stone	250	500
	Food	0	50
# of Enemy Units	Castle	0	2
	defense_tower	2	1
	Worker	0	0
	Swordman	0	0
	Archer	3	2
	Guard	0	0
	Cow	0	0
	battle_machine	0	1
	Armor	30	15
	Sight value	5	2

Table 2: Three_Towers and Destroy_Villag cases similarity calculations

C	*P*	$d_i(P,C)$	$Sim(p_i,c_i)$	w_i	$w_i * Sim(p_i,c_i)$
200	500	0.429	0.571	2.222E-05	1.26984E-05
200	500	0.429	0.571	2.22222E-05	1.26984E-05
250	500	0.333	0.667	0.000032	2.13333E-05
0	50	1	0	0.001	0
0	2	1	0	0.5	0
2	1	0.333	0.667	2	1.333333333
0	0	Discarded	Discarded	Discarded	Discarded
0	0	Discarded	Discarded	Discarded	Discarded
3	2	0.2	0.8	2	1.6
0	0	Discarded	Discarded	Discarded	Discarded
0	0	Discarded	Discarded	Discarded	Discarded
0	1	1	0	2	0
30	15	0.333	0.667	0.009	0.005925926
5	2	0.429	0.571	0.222	0.126984127
Sum				6.732	3.066

- The fourth step is to check the result of the previous similarty equation according to the Real-See algorithm. From which we can see that the Sim(P,C)<=0.5 which means that the received case (defend the castle) similarity to the stored one (the three towers) is week and that the recevied case will be stored in the receiver engine case-base.
- The last step is to pick the next new received case and start over from the first step.

■ **Example 2**: To be sure of the results we had to repeat the previous steps on another new received case called tower_of_souls table (3) and table (4).

Similarity (P,C)= 7.226/10.541=0.686

Table 3: The data set of Three_Towers and Tower_of_Souls cases erepresenting14 features.

Features		*Three_Towers (Case C)*	*Tower_of_Souls (Case P)*
Resources	Gold	200	3000
	Wood	200	300
	Stone	250	1000
	Food	0	60
# of Enemy Units	Castle	0	2
	defense_tower	2	1
	Worker	3	1
	Swordman	1	2
	Archer	2	3
	Guard	1	2
	Cow	0	0
	battle_machine	0	0
	Armor	30	20
	Sight value	5	15

Table 4: Three_Towers and Tower_of_Souls cases similarity calculation

C	*P*	$d_i(P,C)$	$Sim(p_i,c_i)$	w_i	$w_i * Sim(p_i,c_i)$
200	3000	0.875	0.125	2.551E-07	3.189E-08
200	300	0.2	0.8	0.0002	0.0002
250	1000	0.6	0.4	3.555E-06	1.422E-06
0	60	1	0	0.0006	0
0	2	1	0	0.5	0
2	1	0.333	0.667	2	1.333
3	1	0.5	0.5	2	1.6
1	2	0.333	0.667	2	1.333
2	3	0.2	0.8	2	1.6
1	2	0	1	2	1.333
0	0	discarded	discarded	discarded	Discarded
0	0	discarded	discarded	discarded	Discarded
30	20	0.2	0.8	0.02	0.016
5	15	0.5	0.5	0.02	0.01
Sum				10.541	7.226

From table (4) we can see that the Sim(P,C)>0.5 Which means that the received case (tower_of_souls) and the stored one (the three towers) are so similar and that the recevied case will not be stored in the receiver engine case-base till the goal and performance similarities according to our algorithm is checked as following.

- The three towers goal is

  winner (player):-
  Objective ("destroy_towers"),
  towercount (0).

- The tower_of_souls goal is

  winner (player):-
  Objective ("defend_from_attack"),
  unitcount (0),
  towercount (1).

To check the similarity of the cases goals we first need to check the similarity of its parts see table (5).

Table 5: goal similarity calculation

	Three towers goal	*Tower_of_souls goal*	*Similarity(s)*	
P_1	Objective ("destroy_towers")	Objective ("defend_from_attack")	No	0
P_2	Missing	unitcount (0)	discarded	
P_3	towercount (0)	towercount (1)	No	0
P_4	winner (player)	winner (player)	yes	1

After that using equation (9), the MoS value is calculated and then evaluated according to equation (10).

$$MoS = \frac{1}{3}\sum_{i=1}^{3}\{0,0,1\} = 1/3$$

$$\text{Goal similarity} = \begin{cases} \text{Not Similar} & MoS \leq ½ \\ \text{Similar} & MoS > ½ \end{cases} \quad (10)$$

Finally from equation (10) we founded out that the three towers case goal is not similar to the tower_of_souls case goal, but as we mentioned before that the three towers case is similar to the tower_of_souls case. So from all the previous and according to the Real-See algorithm we can conclude that the tower_of_souls case will be stored in the receiver engine case based.

- **Example 3**: to test the last case of Real-See algorithm the performance value comparison, we used a stored case called duel and a new received case called tough_battle in table (6).

Table 6: The data set of duel and tough_battle cases erepresenting14 features.

Features		*Duel (Case C)*	*Tough_battle (Case P)*
Resources	Gold	2000	500
	Wood	300	400
	Stone	1500	1000
	Food	30	60
# of Enemy Units	Castle	0	0
	defense_tower	0	0
	Worker	2	3
	Swordman	2	3
	Archer	0	0
	Guard	1	2
	Cow	0	0
	battle_machine	1	3
	Armor	10	40
	Sight value	15	10

Table 7: Duel and Tough_battle cases similarity calculations

C	*P*	$d_i(P,C)$	$Sim(p_i,c_i)$	w_i	$w_i * Sim(p_i,c_i)$
2000	500	0.6	0.4	8.88889E-07	3.55556E-07
300	400	0.143	0.857	0.0002	0.0002
1500	1000	0.2	0.8	0.000008	0.0000064
30	60	0.333	0.667	0.002	0.002
0	0	Discarded	Discarded	Discarded	Discarded
0	0	Discarded	Discarded	Discarded	Discarded
2	3	0.2	0.8	2	1.6
2	3	0.2	0.8	2	1.6
0	0	Discarded	Discarded	Discarded	Discarded
1	2	0.333	0.667	2	1.333
0	0	discarded	Discarded	Discarded	discarded
1	3	0.5	0.5	0.5	0.25
10	40	0.6	0.4	0.002	0.001
15	10	0.2	0.8	0.08	0.064
Sum				6.585	4.849

Similarity (P,C)=4.849/6.585=0.73654

From table (7) we can see that the Sim(P,C)>0.5 Which means that the received case (tough_battle) and the stored one (duel) are so similar and that the recevied case will not be stored in the receiver engine case-base till the goal and performance similarities according to our algorithm is checked .

As in example 2 we will check the duel and tough_battle cases goal similarities as following:

- The duel goal is
  winner (player):-
  objective (“defend_from_attack”),
  unitcount (0).
- The tough_battle goal is
  winner (player):-
  Objective (“defeat_enemy”),
  unitcount (0).

To check the similarity of the cases goals we first need to check the similarity of its parts see table (8)

Table 8: Goal Similarity Calculation

	Duel Goal	*Tough_Battle Goal*	*Similarity(S)*	
P_1	Objective (“defend_from_attack”)	Objective (“defeat_enemy”)	No	0
P_2	unitcount (0).	unitcount (0).	Yes	1
P_3	winner (player)	winner (player)	Yes	1

After that using equation (9) the MoS value is calculated and then evaluated according to equation (10)

$$\text{MoS} = \frac{1}{3}\sum_{i=1}^{3}\{\mathbf{0},\mathbf{1},\mathbf{1}\} = 2/3$$

From equations (9) and (10) we can see that the duel case goal is similar to the tough_battle case goal, we also mentioned before that the duel case is similar to the tough_battle case. From this and according to Real-See algorithm we can definitely say that we need to check the last case of our algorithm the performance value case.

Suppose that the performance of duel case is 0.63 while the performance of the tough_battle case is 0.7 this means that the received tough_battle case will be stored in the engine case-based. And after storing the case, the case comparator will start over again from the first step with a new received case.

5. Conclusions

In this paper, we have presented an experience exchanging model to improve the performance of RTS game engines through exchanging experiences of facing new un-programmed opponent scenarios. Our model is based on the game case-based reasoning system specially on adding new cases to it. New cases are sent by other engines that faced new opponent scenarios and beat them to help the engine dealing with these scenarios if it faces them in the future. We believe this will reveal game players from downloading a new engine of the game and loosing their saved stages.

Our main priority here was to be sure that these received cases are all new to the system and have no matching cases in the game CBR. In order to do that we also introduced an algorithm which we call Real-See to check the similarity of these received cases to the stored ones. This algorithm is not concentrating on the case description only but on the case goal and performance too. We tested the Real-See algorithm on real picked cases from 3D-Glest RTS game and it performed well.

Future Work

In the future we plan to pursue several future researches on the case-based situation assessment depending on Real-See algorithm and whether it helps to enlarge the case-based or to shrink it. We also will try to introduce an implementation of the Real-See algorithm in both Glest and Waragus open-source real time strategy games.

References

[1] Aamodt A. and Plaza E.,"Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches" In Proceedings of AICom - Artificial Intelligence Communications, IOS Press, Vol. 7: 1,1994, pp. 39-59

[2] Aha D., Molineaux M. and Ponsen M., "Learning to win: Case-based plan selection in a real-time strategy game". In Proceedings of Sixth International Conference on Case-Based Reasoning , 2005,pp. 5-20. Springer.

[3] Balla R. and Fern A., "UCT for Tactical Assault Planning in Real-Time Strategy Games". In Proceedings of the 21st international JONT conference on Artificial intelligence, 2009,pp.40-45, Pasadena, California, USA.

[4] Copeland, J.,"Artificial intelligence: A Philosophical Introduction". United Kingdom: Blackwell Publishers,1993.

[5] Dimitriadis V. K., "Reinforcement Learning in Real Time Strategy Games Case Study on the Free Software Game Glest". Department of Electronic and Computer Engineering Technical University of Crete.China,2009.

[6] Fasciano M., "Everyday-World Plan Use". The University of Chicago, Computer Science Department.Chicago, Illinois,1996.

[7] Goodman M.,"Results on Controlling Action With Projective Visualization". In Proceedings of the Twelfth National Conference on AI. 1994,pp. 1245-1250. Seattle, WA: AAAI Press.

[8] Hammond K.," Case-BasedPlanning: AFramework for Planning from Experience". In Proceedings of Journal of Cognitive Science. Ablex Publishing, Norwood, NJ. Vol. 14,1994.

[9] Kok, E. "Adaptive reinforcement learning agents in RTS games". University Utrecht, The Netherlands,2008.

[10] Li M., et.al.,"The Similarity Metric". In Proceedings of IEEE Transactions on Information Theory,Aug. 2004.

[11] Marthi B, Russell S., and Latham D.,"Writing Stratagus Playing Agents in Concurrent Alisp". In Proceedings of Workshop on Reasoning Representation and Learning in Computer Games (IJCAI-05), 2005,pp. 67-71.

[12] Ponsen M., Muñoz-Avila H., Spronck P., and Aha D., " Automatically Generating Game Tactics via Evolutionary Learning", Proceedings of AI Magazine, vol.(27),2006,pp.75-84.

[13] Simpson R.," A Computer Model of Case-based Reasoning in Problem Solving". PhD thesis, Georgea Institute of Technology,1985.

[14] Ponsen M. and Spronck P.," Automatically acquiring domain knowledge for adaptive AI games using evolutionary learning". In Proceedings of the 17th conference on Innovative applications of artificial intelligence. Vol.(3) .Pittsburgh, Pennsylvania, 2004,pp:1535-1540.

[15] Santiago O., Mishra K., Sugandh N. and Ram A., "Case-based planning and execution for real-time strategy games". In Proceedings of ICCBR-2007, 2007,pp 164-178.

[16] Tracy B.,"Game Intelligence AI Plays Along". In Proceedings of the Computer Power User. Volume 2, Issue 1,2002, pp 56-60.

[17] Ulam P., Goel A. & Jones J.,"Reflection in Action: Model-Based Self-Adaptation in Game Playing Agents". In Proceedings of D. Fu & J. Orkin (Eds.) Challenges in Game Artificial Intelligence: Papers from the AAAI Workshop (TR WS-04-04). San Jose, CA: AAAI Press,2004.

[18] WWW.Wikipedia.org

Mostafa Aref is a Professor and Chairman of Computer Science Department, Ain Shams University. He got his B.Sc. from Ain Shams University, Egypt; his M.Sc. from University of Saskatchewan, Canada; and his Ph.D. from University of Toledo, Ohio, USA. He worked in Several Universities in USA, Saudi Arabia and Egypt. Currently he is a coordinator of two research groups on NLP and RTS games in Ain Shams University.

Magdy Zakaria is an assistant professor and Chairman of Computer Science Department in Mansoura University. He is the Decision Support Systems unit coordinator at faculty of computers &Information in Mansoura University. He has supervised over 10 PhDs and 15 masters mostly specialized in Artificial Intelligence and its applications related to real life. As a result of his work he has published over 40 papers. Current project is grid computing.

Shahenda Sarhan is a PHD student and an assistant lecturer at the Computer Science Department in Mansoura University. Her subject is in Real-Time Strategy games.