In 9th European Conference on Case-Based Reasoning (ECCBR 2009), Trier, Germany

Situation Assessment for Plan Retrieval in Real-Time Strategy Games

Kinshuk Mishra, Santiago Ontañón, and Ashwin Ram

Cognitive Computing Lab (CCL)
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332/0280
{kinshuk,santi,ashwin}@cc.gatech.edu

Abstract. Case-Based Planning (CBP) is an effective technique for solving planning problems that has the potential to reduce the computational complexity of the generative planning approaches [8, 3]. However, the success of plan execution using CBP depends highly on the selection of a correct plan; especially when the case-base of plans is extensive. In this paper we introduce the concept of a *situation* and explain a *situation assessment* algorithm which improves plan retrieval for CBP. We have applied situation assessment to our previous CBP system, Darmok [11], in the domain of real-time strategy games. During Darmok's execution using situation assessment, the high-level representation of the game state i.e. situation is predicted using a decision tree based Situation-Classification model. Situation predicted is further used for the selection of relevant knowledge intensive features, which are derived from the basic representation of the game state, to compute the similarity of cases with the current problem. The feature selection performed here is knowledge based and improves the performance of similarity measurements during plan retrieval. The instantiation of the situation assessment algorithm to Darmok gave us promising results for plan retrieval within the real-time constraints.

1 Introduction

Generative planning techniques are typically inapplicable for solving problems with extensive search spaces within real-time constraints. Case-based planning (CBP) [13] has the potential of reducing the computational complexity of traditional planning techniques. Specifically, CBP works by reusing previous stored plans for new situations instead of planning from scratch. Thus, CBP is a promising paradigm to deal with real-time domains. In this paper we will focus in Darmok, [11] a case-based planning system that is able to deal with the complexity of real-time strategy (RTS) games. Darmok was designed to play WARGUS, an open source implementation of the famous Warcraft II. However, the success of plan execution using CBP in such domains depends on the quality of plan selection within the real-time constraints. The performance of Darmok's plan

retrieval suffers when the case-base stores numerous plans representing several strategies played over maps of different sizes and terrain formations. In this paper we explain our work on situation assessment technique applied to Darmok for better plan retrieval in real-time.

A *Situation* is a high-level representation of the state of the world. For example, in the WARGUS domain the player might be in an *attacking* situation, or in a *base development* situation, among others. Depending on which situation the player is in, different aspects of the world state will be important to take decisions. Thus, in order to select which strategy to execute, it is important to know the current situation. Situations can be predicted based on raw features that can be directly computed from the game state, i.e. *shallow features*. However, shallow features by themselves are not strong enough for selection of a strategy in a game. Additional derived *deep features* for a situation are needed. For example, shallow features, like the ratio of a player's resources to that of the opponent, by themselves are less suggestive of usefulness of an appropriate strategy. However deeper features, like knowing the existence of path or a barrier between the player and its opponent, can help in choosing a rush or a tunneling strategy. Situation assessment is used to predict the situation of a game state based on the shallow features. This information is used to further select a set of deep features specific to the situation for choosing the best strategy. Formally, *Situation Assessment* is a process of gathering high-level information using low-level data to help in better decision making.

Our general situation assessment technique comprises of four steps: *shallow feature selection*, *model generation*, *model execution* and *case retrieval*. Firstly, a subset of shallow features is selected which are used for classification of a game state into a situation. Then three models: a) for classification of game state into situations based on shallow features, b) for mapping of situations to cases and c) for mapping of situations to deep features respectively are generated. Execution of these models helps Darmok to classify a game state into a situation and then retrieve the most optimal plan using situation specific deep features. Plan retrieval results in Darmok using situation assessment have been promising.

The rest of the paper is organized as follows. Section 2 presents a summary of the related work. Then, Section 3 briefly explains the architecture of the Darmok system. After that, Section 4 describes the process of situation assessment. Section 5 explains the situation assessment algorithm applied to Darmok System. Section 6 provides an illustration of the process. Finally, we summarize our experiment results in Section 7 and then end with a conclusions section.

2 Related Work

There are several relevant areas of work related to our approach, namely: situation assessment, feature selection, and the application of CBR to computer game AI. Concerning situation assessment, work has been done extensively in the area of information fusion [4] and defense related command and control projects [2], however little work has been done using CBR. Kolodner [10] defined situation

assessment as the process of deriving additional features in a particular situation in order to compare it with previous experiences, but no CBR system to our knowledge implements such process. Kofod-Petersen and Aamodt [9] define a case-based situation assessment system for a mobile context-aware application. The system uses case-based reasoning to determine the situation in which the user might be in, and the possible goals associated with these situations. They define a situation as a context, and define a hierarchy of contexts in which the user might be in. The difference with our work is that we are interested in situation assessment as a way to select a subset of features that allows us to perform better case retrieval.

Plenty of work exists on feature selection in the machine learning literature. Hall and Holmes [7] present a nice overview and empirical evaluation of several feature selection techniques. Some well-known techniques include: information-gain based techniques [14], Principal Component Analysis, Correlation-based Feature selection [6], or Cross-validation methods (that simply run the learning algorithm repeatedly with different feature subsets and select the best one empirically). The main difference of our work with the existing feature selection techniques, is that the set of possible features from where we can select features is too large and the examples are few, and thus we need a more knowledge-based feature selection method (situation assessment) that does not involve trying feature-by-feature.

Concerning the application of case-based reasoning techniques to computer games, Aha et al. [1] developed a case-based plan selection technique that learns how to select an appropriate strategy for each particular situation in the game of WARGUS. In their work, they have a library of previously encoded strategies, and the system learns which one of them is better for each game phase. In addition, they perform an interesting analysis on the complexity of real-time strategy games (focusing on WARGUS in particular). Another application of case-based reasoning to real-time strategy games is that of Sharma et al. [12], where they present a hybrid case-based reinforcement learning approach able to learn which are the best actions to apply in each situation (from a set of high level actions). The main difference between their work and ours is that they learn a case selection policy, while our system constructs plans from the individual cases it has in the case-base. Moreover, our architecture automatically extracts the plans from observing a human rather than having them coded in advance.

3 Case-Based Planning and Execution in Wargus

In this section we will present an overview of WARGUS and of the Darmok system. WARGUS (Figure 1) is a real-time strategy game where each player's goal is to remain alive after destroying the rest of the players. Each player has a series of troops and buildings and gathers resources (gold, wood and oil) in order to produce more troops and buildings. Buildings are required to produce more advanced troops, and troops are required to attack the enemy. In addition,

Fig. 1. A screenshot of the WARGUS game.

players can also build defensive buildings such as walls and towers. Therefore, WARGUS involves complex reasoning to determine where, when and which buildings and troops to build. For example, the map shown in Figure 1 is a 2-player version of the classical map "Nowhere to run nowhere to hide", with a wall of trees that separates the players. This maps leads to complex strategic reasoning, such as building long range units (such as catapults or ballistae) to attack the other player before the wall of trees has been destroyed, or tunneling early in the game through the wall of trees trying to catch the enemy by surprise.

The Darmok system [11] is a case-based planning system designed to play the game of WARGUS. Darmok learns plans (cases) by observing a human playing the game, and then reuses such plans combining and adapting them to play new games using case-based planning methods. Figure 2 presents the Darmok architecture that is split into two main processes: *Behavior Acquisition* and *Behavior Execution*. The *Behavior Acquisition* process is performed by the Revision and Case Learning modules in the following way. Each time a human plays a game, a trace is generated (containing the list of actions performed by the human). During revision, the human annotates that trace stating which goals he was pursuing with each action. This annotated trace is processed by the case learning module that extracts plans in form of cases from the trace. Each plan consists of two components:

- A Behavior: consisting of a goal and a plan. Basically, a behavior stores that, to achieve a particular goal, the human used a particular plan.
- An Episode: consisting of a reference to a behavior, a game state and an outcome. Episodes store how well a particular behavior performed in a particular game state. The outcome is a real number between 0 and 1, stating how much the behavior achieved its goal in the specified game state.

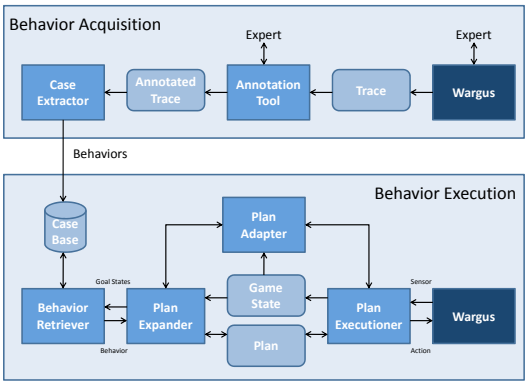

Fig. 2. Overview of our case-based planning approach.

Thus, the case-base of Darmok is composed of behaviors, and each behavior is associated with a bunch of episodes. Behaviors are learnt from traces, and episodes can be learnt either from traces or from experience. The *Behavior Execution* process is performed by the rest of the modules in the architecture, and works as follows. Darmok starts off by giving the initial unexpanded goal "Win Wargus" to the Plan Expander. Each time the Plan Expander wants to expand a goal, it asks the Behavior Retriever for a behavior, which uses the case-base to select the best behavior for the goal at hand in the current game state. The Plan Adapter adapts the retrieved behaviors before they are inserted in the current plan, which is maintained by Darmok for execution. The Plan Executioner constantly tries to execute that plan that might be only partially expanded. Such a plan is maintained by the Plan Expander that looks for unexpanded goals in the plan and tries to expand them. Thus, the Plan Executioner tries to see if there is any part of the plan that has been expanded to the level of primitive actions that can be sent to the game, and executes such actions if possible. Some of the primitive actions in the game are like *move-unit*, *repair-building*, etc.

Finally, notice that WARGUS is a dynamic domain, thus the game state changes constantly. For that reason, the Plan Expander delays the adaptation of plans till the last moment (right before they have to start execution) to ensure they are adapted with the most up-to-date game state (Delayed Adaptation). In this paper we will focus on the behavior retrieval problem. See [11] for a detailed explanation of the Darmok system.

4 Situation Assessment for Case Retrieval

Traditionally, in Case-Based Reasoning the process of case retrieval is done by selecting the case from the case-base that has the closest similarity to the world state of the problem. This similarity is measured over the various features that are computed from the representation of the world state. The key to the most

optimal case selection is choosing the set of most important features which improve the similarity measurement during the case selection process. The choice of features depends on their relevance in representing the high-level inferential knowledge about the world state. Here, we define a *Situation* as a high-level representation of the state of the world in a problem. For example, in the WARGUS domain the player might be in an *attacking* situation, or in a *base building* situation among others. Depending on which situation the player is in, different aspects of the world state will be important for decision making. Hence, the task of choosing the most relevant set of features for optimal case selection depends on the current situation of the world. Situations can be predicted based on the raw features that are directly computed from the world state i.e. *shallow features*. These shallow features are generally computationally inexpensive but lack the high-level inferential knowledge about the world. For instance, in the WARGUS domain, the features like ratio of player's gold resources versus that of the opponent or the number of trees in the map are shallow features. Once a situation is predicted, through situation assessment, the additional derived *deep features* specific to a situation are used for comparing the high-level knowledge represented by each case. For instance, in the WARGUS domain, the deep features like knowing the existence of path or a barrier between the player and its opponent, can help in choosing a rush or a tunneling strategy. The deep features are generally computationally expensive but provide information very relevant for case selection in specific situations. As we said before, situation assessment is a process of gathering high-level information using low-level data to help in better decision making. Thus, in the case of CBR, it is the process of gathering the important features and other pieces of information that will help us retrieve the most appropriate case.

Our general situation assessment algorithm is described in Figure 3. It comprises of four main steps:

- **Shallow Feature Selection**: During this first step, a situation annotated trace T is provided to a feature selection algorithm. An annotated trace consists of a sequence of world states annotated with the set of shallow features computed for each world state and the appropriate situation that world state corresponds to. This algorithm returns the set of shallow features F'_s which have high information gain. Specifically, in Darmok, we have used best-first greedy hill-climbing algorithm [5] for filtering the high information gain shallow features.
- **Model Generation**: In this step the following three models are generated:
  - The *Situation-Classification Model*, M_{cf}, is built by providing F'_s and T to a classification algorithm. This model is useful for classification of a world state to a situation using shallow features in F'_s. In Darmok, we have used a standard algorithm inducing a decision tree classifier model.
  - The *Situation-Case Model*, M_c, provides a mapping from the set of situations S to a subset of cases in the case-base C. It can be built using statistical or empirical analysis. This model captures the intuition that not all the cases will be useful in all the situations.

Function SituationAssessment(C, T)
1 Shallow Feature Selection
$F'_s = \text{SelectShallowFeatures}(F_s, T)$

2 Model Generation
$M_{cf} = \text{GenerateClassificationModel}(F'_s, T)$
$M_c = \text{GenerateCaseModel}(S, C)$
$M_{df} = \text{GenerateDeepFeatureModel}(S, F_d)$

3 Model Execution
$s = \text{GetCurrentSituation}(M_{cf}, F'_s)$
$C' = \text{GetRelevantCaseSubset}(M_c, s)$
$F'_d = \text{GetDeepFeatureSet}(M_{df}, C')$

4 Case Retrieval
Return RetrieveCase(C', F'_d, F_s)
End-Function

Fig. 3. General Situation Assessment Algorithm. Where C is the case-base, T is the situation annotated training set. F_s and F_d are the set of all shallow and deep features respectively. F'_s is the subset of high information gain features selected from F_s. M_{cf} is the Situation-Classification model. S is the universal set of all possible situations. M_c and M_{df} are the Situation-Case model and Situation-Deepfeature models respectively, built empirically in Darmok. s represents the current situation of the game state. C' is the most relevant subset of cases from the case-base obtained for s from the execution of M_c. F'_d is the subset of deep features obtained from execution of M_{df}. RetrieveCase returns the best case from C' using F'_d and F_s.

  - The *Situation-Deepfeature Model*, M_{df}, provides a mapping from S to deep features in the set F_d. This mapping is done using a feature selection algorithm or by using empirical knowledge.
- **Model Execution**: In this third step, the models generated in the previous step are executed to get the current situation s, the subset of cases C' from the case-base C and the subset of deep features F'_d which are most relevant to s. s is obtained by running M_{cf} over F'_s. Once s is known, using M_c and M_{df}, C' and F'_d are obtained respectively.
- **Case Retrieval**: This is the last step where using F'_d and F_s the most similar case in retrieved from C' using normal retrieval techniques.

5 Situation Assessment applied to Darmok

In this section we shall present the instantiation of the situation assessment algorithm in our system, Darmok, to improve the performance of its case-based plan retrieval. We apply the General Situation Assessment algorithm in Figure 3 to Darmok, but split in two stages:

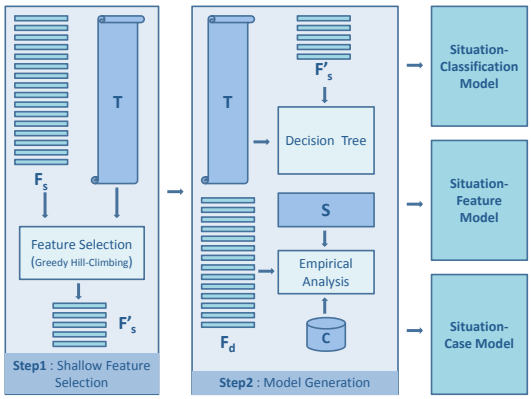

Fig. 4. Offline Stage of Situation Assessment. Where T is the trace, F_s is the set of shallow features, F'_s is the subset of shallow features after feature selection, F_d is the set of deep features, S is the set of situations and C is the case-base.

Table 1. Goal to Situation Mapping

Goals	Situations
ResearchGoal	Base Development, Defense, Dev-Defense
AbsoluteBuildUnitsGoal, RelativeBuildUnitsGoal	Base Development, Defense, Attack, Dev-Defense, Dev-Attack
ResourceInfrastructureGoal	Base Development
KillAllUnitsOfTypeGoal, KillUnitGoal, DefeatPlayerGoal	Attack
WinWargusGoal	Beginning

- The Offline Stage: comprising of Feature Selection and Model Generation before the game-play.
- The Online Stage: comprising of Model Execution and Plan Retrieval during the game-play.

We perform situation assessment in two stages since the models required for predicting the situation during Darmok's game-play are built just once at the start. Therefore, the models can be easily generated offline using standard feature selection and classification algorithms.

5.1 Offline Stage

As shown in the Figure 4 the offline stage of the situation assessment algorithm in Darmok consists of the first two steps of the algorithm from Figure 3.

In the first step of shallow feature selection the set of shallow features F_s^w in WARGUS and the situation annotated trace T^w are provided to the best-first greedy hill-climbing-with-backtacking feature selection algorithm [5]. T^w is generated over various game states, with the values of all the shallow features, by forcing Darmok to play particular maps with the best strategies for those maps, which were demonstrated by an expert. In our experiments, trace generation and annotation was automated, based on the goals that Darmok was pursuing in those particular game states, because a goal being pursued is the high-level representation of the game state that helps in choosing a particular plan.

The feature selection algorithm we have used returns the set of shallow features $F_s^{w'}$, which have high information gain. Once $F_s^{w'}$ is generated it is provided along with T^w to a pruning enabled C4.5 decision tree algorithm [5] to learn a decision tree situation-classifier model M_{cf}^w. M_{cf}^w generated here is used in real-time during game-play for predicting the situation of the game state.

The Situation-Case model M_c^w is built using empirical knowledge of the WARGUS domain. The cases are mapped manually to situations based on the goal of the behavior in the plan represented in the case as shown in Table 1.

The Situation-Deepfeature model M_{df}^w is constructed manually by using an expert's empirical knowledge about usefulness of various deep features in certain situations. For example, using deep features like *attacking-speed-of-troops* and *attacking-radius-of-troop-formation* are more relevant in choosing a strategy while a player is in *attacking* situation as compared to when he is in *base development* situation.

5.2 Online Stage

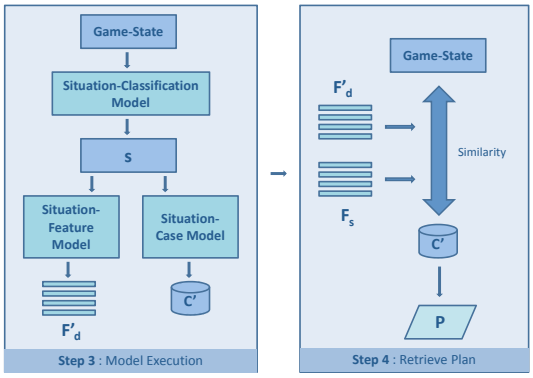

Fig. 5. Online Stage of Situation Assessment. Where s is the current situation, F_d' is the relevant subset of deep features, C' is the relevant subset of the case-base, F_s is the set of shallow features and P is the retrieved plan.

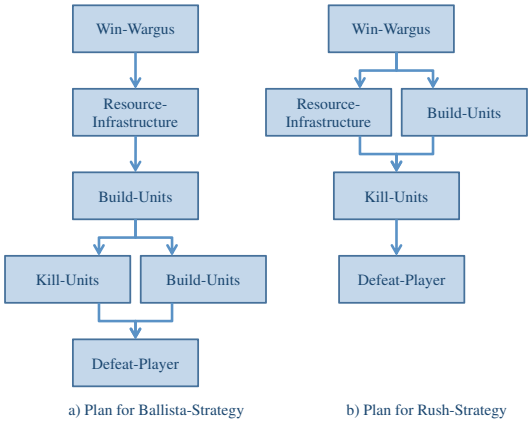

Fig. 6. Plans for ballista and rush strategies for game-play in WARGUS. a) The ballista strategy comprises of initial resources development followed by building the units to strengthen the player base. Later the units are built and sent to kill the opponent units in parallel. b) The rush strategy comprises of quick resource development and building the units at the start in parallel followed by killing the opponent units.

This stage comprises of the last two steps of the algorithm of Figure 3 as shown in Figure 5. This stage in interleaved with the case-based planning and execution of Darmok. During Darmok's online game-play when the Plan Expander requests the Behavior Retriever for a new plan, Darmok, before the plan retrieval, first executes the model M^w_{cf} followed by the parallel execution of M^w_c and M^w_{df} respectively . Darmok computes the value of $F^{w'}_s$ from the current game state and evaluates the current situation through the decision tree based M^w_{cf}.

Once the current situation s is evaluated, mapping based M^w_c and M^w_{df} model suggest the case-base subset $C^{w'}$ and a deep feature subset $F^{w'}_d$ for the final plan retrieval.

In the last step of this stage the features in $F^{w'}_d$ and F^w_s are used to measure the similarity of the cases in $C^{w'}$ and return the plan having most similar goal and game state to the scenario during the game-play. The similarity is measured by placing more importance to the features in $F^{w'}_d$ as compared to F^w_s.

Let us illustrate this process with an example.

6 Example

Let us illustrate the online stage of the situation assessment process in Darmok with an example. Imagine that Darmok has just started the game-play against the built in game AI opponent in a variation of the well-known map "Nowhere to run nowhere to hide" (NWTR) as shown in Figure 1. Unlike the typical NWTR maps that have a wall of trees separating the opponent this map has a narrow opening in the wall of trees. Darmok starts the execution with the initial goal of

"WinWargus". During the execution, the Plan Expander requests the Behavior Retriever to return a plan to satisfy this goal. Here, the online stage of situation assessment gets triggered and Darmok uses the decision tree based situation-classifier and the set of relevant shallow features, say, *lumber* (number of trees in the map), *food* (amount of food), *gold* (amount of gold of the player), *peasants* (number of peasants) and *units* (number of units the player has), which were chosen during the offline stage, to predict the current situation as *beginning*.

Once the situation is predicted the Situation-Case model, essentially a mapping of situations to plans based on the goals that the plans satisfy as shown in Table 1, is searched to find the subset of the plans which are relevant to the beginning situation. Since the beginning situation is mapped to plans satisfying the "WinWargus" goal, Darmok has successfully narrowed down its search space to the set of few relevant plans.

Darmok also refers to the Situation-Deepfeature model to get the set of most relevant features for the beginning situation: *ispath* (a boolean feature that is true when there is a path from the player base to the enemy base), *wallbarrier-width* (the width of the biggest barrier between the player and the enemy) and *baseproximitydistance* (distance between the player's base and the enemy base).

Beginning situation is where player has to choose a game strategy which is most optimal for a particular map-terrain and opponent strength, to win the game. Assuming that there are just two plans as shown in Figure 6 for the beginning situation in the reduced case-base $C^{w'}$, each representing different game strategies, the task of the Darmok's Behavior Retriever is to choose the plan with the best strategy for the current game state.

The two plans in Figure 6 represent the *ballista* and *rush* strategies. Ballista strategy is good for maps where the player and the opponent are separated by a wall of trees while rush strategy is good when there is a path from player to the opponent in the map. The selection amongst these strategies depends highly on the measurement of the deep features like *ispath*, *wallbarrierwidth* and *baseproximitydistance* since the concept of existence of path between player and opponent bases, wall of trees and separation distance of base are not expressed through shallow features like gold, lumber, trees, etc. Also the other deep features like *attacking-speed-of-troops* and *attacking-radius-of-troop-formation* are more relevant for attack strategy selection rather than for game strategy selection and hence are not be considered for beginning situation. Using the three deep features and all the shallow features in a weighted manner Darmok's Behavior Retriever searches the two plans and retrieves the plan for the rush strategy since the game state's similarity is found to be higher for the episodes of plan representing rush strategy.

If no deep features were used, Darmok would have difficulty identifying which strategy to pick. Moreover, if no situation assessment is used, and all the deep features are always used for retrieval, the time consumed to compute all the deep features would be prohibitive (as we will show in the next section).

7 Experimental Evaluation

In our evaluation we found out that the performance of plan retrieval applying situation assessment algorithm (Figure 3) is better compared to plan retrieval without the application of situation assessment by conducting three set of experiments as follows:

- Exp_1: Darmok performed plan retrieval without the situation assessment algorithm. Darmok used only the *shallow* features for computing similarity during the retrieval stage and no situation prediction was performed.
- Exp_2: Darmok performed plan retrieval without the situation assessment algorithm. Darmok used only the *deep* features for computing similarity during the retrieval stage and no situation prediction was performed.
- Exp_3: Darmok performed plan retrieval using the situation assessment algorithm. It used selected shallow and deep features for similarity computation during the retrieval stage.

The experiments Exp_1, Exp_2 and Exp_3 were conducted over 11 variations of the "Nowhere to run nowhere to hide" (NWTR) map (with a wall of trees separating the opponents that introduces a highly strategic component in the game) and over "Garden of War" (GOW) map (large map having lot of open spaces, with tree and gold resources in the middle). Darmok was tested with 10 different strategies with slight variations, demonstrated over 6 out of the 11 different maps. The 10 strategies demonstrated were variations of the *ranged attack* (ballistas attack over the wall of trees), *rush* (footmen are built and quickly sent to attack the opponents when there is a path betweem them), *tunneling* (footmen and knights are built and tunnel through the wall of trees to attack the opponent) and *towering* (towers are built around the wall of trees to block the enemy).

We conducted the experiments with 10 traces in the case-base (that gives a total of 52 behaviors and 52 episodes in the case base) over 5 runs of the game and measured the performance of Darmok's plan retrieval in Exp_1 and Exp_3 over the following parameters: 1) number of wins, 2) number of draws, 3) number of losses, 4) player's score assigned by WARGUS, and 5) opponent's score assigned by WARGUS. We also report the average retrieval time for each plan by Darmok. For our experiments, and in order to properly validate retrieval, and retrieval only, episode learning and structural plan adaptation were disabled in Darmok.

Tables 2 and 3 show the results of Exp_1 and Exp_3 respectively. The first column shows the map in which a game was played, the next three columns show the number of wins, draws and losses respectively. The last two columns show the scores of player and the opponent which are assigned by WARGUS. The bottom row of each table shows a summarized view of Darmok's win ratio and average score ratio, where the win ratio is the number of wins divided by the total number of games played and the average score ratio is the average score of the player divided by the opponent's average score. As seen, there is a clear improvement in the results of the game-play in Exp_3; with win ratio of

map	*win*	*draw*	*loss*	*player score*	*opponent score*
NWTR1	1	0	4	1068	1331
NWTR2	3	1	1	2410	562
NWTR3	2	0	3	2094	1613
NWTR4	1	0	4	1964	1791
NWTR5	1	0	4	1296	1700
NWTR6	1	1	3	1652	1128
NWTR7	1	0	4	1016	2161
NWTR8	2	0	3	1418	1560
NWTR9	0	0	5	832	2643
NWTR10	0	0	5	406	1997
NWTR11	0	0	5	82	1507
GoW	2	0	3	756	626
Win Ratio	**0.233**			Average Score Ratio	**0.81**

Table 2. Exp_1 results.

map	*win*	*draw*	*loss*	*player score*	*opponent score*
NWTR1	2	3	0	7136	1386
NWTR2	5	0	0	3000	24
NWTR3	5	0	0	1800	0
NWTR4	4	0	1	1180	388
NWTR5	2	0	3	1794	1505
NWTR6	5	0	0	2450	50
NWTR7	5	0	0	3100	94
NWTR8	0	0	5	1750	2790
NWTR9	5	0	0	3356	60
NWTR10	5	0	0	1410	50
NWTR11	0	0	5	4466	3601
GoW	3	1	1	1355	585
Win Ratio	**0.683**			Average Score Ratio	**3.12**

Table 3. Exp_3 results.

0.683 which is thrice better than win ratio of 0.233 in Exp_1. Also, the average score ratio in case of Exp_3 is four times better than Exp_1 (i.e. 3.12 as compared to 0.81), which indicates that Darmok wins convincingly and even its losses are well-fought. Situation assessment through its results, thus, can be seen to increase the performance of plan retrieval. An interesting observation is that on maps NWTR3, NWTR6, NWTR7, NWTR10 for which the expert demonstrated strategies, Darmok won on all 5 occassions in Exp_3. For the same maps in Exp_1, Darmok had marginal success and even complete failure in case of map NWTR10. In general Darmok performs better using situation assessment over all the maps except NWTR8 and NWTR11. For the map NWTR8, Darmok's performance in Exp_1 is marginally better compared to the performance in Exp_3. Darmok's performance suffers on map NWTR11 even after correctly retrieving the plan, demonstrated for map NWTR11 by the expert, from the case-base since in adversarial non-deterministic domain like WARGUS there are lots of factors which can influence winning a game. On the map NWTR11, Darmok's win ratio is same in Exp_1 and Exp_3, however, using situation assessment improved the average score ratio in Exp_3.

Using selective deep features like *ispath*, *wallbarrierwidth*, etc. based on the situations certainly improved the similarity metric for proper retrieval. Experiments were also conducted to measure the average plan retrieval time in seconds with all the deep features as shown in Table 4. Exp_2 was simply not feasible

	Exp_1	Exp_2	Exp_3
retrieval time	0.016	46.428	4.990

Table 4. Average retrieval time (in seconds) for Exp_1, Exp_2 and Exp_3.

and experiments couldn't be run, retrieval time was some times over a minute, completely inappropriate for the dynamic nature of WARGUS. Using situation assessment we managed to reduce the time to a few seconds, which is acceptable for the speed at which WARGUS is played (notice that retrieval is only executed a few times during game-play, and thus spending a few seconds on selecting the appropriate plan paid off, as shown above). Interestingly, it was observed that using all the deep features makes the system use lots of irrelevant information during plan retrieval and reduces the efficiency. It is therefore necessary to filter the deep features and select only the relevant ones. Also, Situation assessment reduced the retrieval time by ten times through use of situation relevant deep features. The above observations indicate that deep features improve the retrieval performance only if chosen appropriately.

In the experiments conducted, Darmok's score in Exp_3 increased to 0.683 compared to 0.233 in Exp_1. The retrieval times with application of the situation assessment algorithm are also acceptable for Darmok's real-time performance and clearly show that quality of plan retrieval has improved.

8 Conclusions

In this paper we have presented situation assessment technique for plan retrieval in real-time strategy games. Our technique is a knowledge based approach for feature selection for improving the performance of case retrieval in case-based reasoning systems. Situation assessment essentially involves two major steps before case retrieval: the generation of models for case-base size reduction and feature selection and then their execution to get the reduced size case-base and set of high information features for case selection. The main characteristics of our approach are a) the capability to perform a knowledge based feature selection rather than a feature by feature, b) the ability to perform search in the case-base in a fast and focused manner by reducing the search space to the set of relevant cases using computationally inexpensive features, c) the capability to resize the dimensions of similarity metric based on the high-level representation of the game state i.e. situations. We have implemented the situation assessment algorithm inside the Darmok system that plays the game of WARGUS. The experiments conducted using situation assessment show a great improvement of performance in the system.

The main contributions of our technique are: 1) introduction of a domain independent situation assessment algorithm that can be applied for knowledge based feature selection to any domain; 2) the idea of case-base size reduction during the search operation through Situation-Case mapping; 3) the introduction

of the concept of a situation as a high-level game state representation for effective plan selection for game strategies; 4) the idea of selective similarity-metric resizing based on the game state situation.

As future lines of work, we plan to explore strategies to fully automate the situation assessment procedure. Currently, the Situation-Case and the Situation-Deepfeature models are empirically determined by hand. Also, the subset of situations is defined by hand. Automated techniques to generate such models will greatly increase the applicability of the approach.

References

1. David Aha, Matthew Molineaux, and Marc Ponsen. Learning to win: Case-based plan selection in a real-time strategy game. In *ICCBR'2005*, number 3620 in LNCS, pages 5–20. Springer-Verlag, 2005.
2. Robert P. Arritt and Roy M. Turner. Situation assessment for autonomous underwater vehicles using a priori contextual knowledge. In *13th International Symposium on Unmanned Untethered Submersible Technology (UUST)*, 2003.
3. Ralph Bergmann, Hector Muñoz-Avila, Manuela M. Veloso, and Erica Melis. Cbr applied to planning. In *Case-Based Reasoning Technology*, pages 169–200, 1998.
4. E. Blasch, I. Kadar, J. Salerno, M. M. Kokar, S. Das, G. M. Powell, D. D. Corkill, and E. H. Ruspini. Issues and challenges of knowledge representation and reasoning methods in situation assessment (level 2 fusion). *Proc. SPIE 6235*, 2006.
5. Eibe Frank, Mark A. Hall, Geoffrey Holmes, Richard Kirkby, and Bernhard Pfahringer. Weka - a machine learning workbench for data mining. In *The Data Mining and Knowledge Discovery Handbook*, pages 1305–1314, 2005.
6. Mark A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *ICML*, pages 359–366, 2000.
7. Mark A. Hall and Geoffrey Holmes. Benchmarking attribute selection techniques for discrete class data mining. *IEEE Trans. Knowl. Data Eng.*, 15(6):1437–1447, 2003.
8. Kristian F. Hammond. Case based planning: A framework for planning from experience. *Cognitive Science*, 14(3):385–443, 1990.
9. Anders Kofod-Pedersen and Agnar Aamodt. Case-based situation assessment in a mobile context-aware system. In *Artificial Intelligence in Mobile System (AIMS 2003)*, pages 41–49, 2003.
10. Janet Kolodner. *Case-based reasoning*. Morgan Kaufmann, 1993.
11. Santi Ontañón, Kinshuk Mishra, Neha Sugandh, and Ashwin Ram. Case-based planning and execution for real-time strategy games. In *Proceedings of ICCBR-2007*, pages 164–178, 2007.
12. Manu Sharma, Michael Homes, Juan Santamaria, Arya Irani, Charles Isbell, and Ashwin Ram. Transfer learning in real time strategy games using hybrid CBR/RL. In *IJCAI'2007*, page to appear. Morgan Kaufmann, 2007.
13. L. Spalazzi. A survey on case-based planning. *Artificial Intelligence Review*, 16(1):3–36, 2001.
14. Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *ICML*, pages 412–420, 1997.