Discovering abstract concepts to aid cross-environment transfer for a learning agent

Cédric Herpson
Computer Science Laboratory
Paris 6 University
104, av Président Kennedy
75016 Paris, France
00 33.1.44.27.53.90
cedric.herpson@lip6.fr

Vincent Corruble
Computer Science Laboratory
Paris 6 University
104, av Président Kennedy
75016 Paris, France
00 33.1.44.27.72.07
vincent.corruble@lip6.fr

ABSTRACT

Intelligent behavior requires the capacity to apply knowledge in a context different than the one in which it was learned. Though this question has been addressed in a number of domains, within and beyond artificial intelligence, it is still an open research question within the area of learning agents, and is crucial in a number of application domains such as strategy games or military simulations.

In this paper, we address more specifically the issue of transfer of knowledge acquired through online learning, in an environment characterized by its 2D geographical configuration. We propose an autonomous agent architecture that learns from a given map and is then able to improve its performances on another map, through the discovery of relevant abstract concepts which are map-independent. This is achieved through the combination of an agent-centered representation and the supervised and unsupervised learning of discriminating features from the environment.

Our architecture is evaluated experimentally on a grid-world environment where two agents duel each other. Results show that the agent's performances are improved through learning, even when it is tested on a map it has not yet seen.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning – *analogies, concept learning, knowledge acquisition.*

General Terms

Algorithms, Performance, Experimentation, Theory

Keywords

Generalization, Abstraction, Agent, Automatic concept learning, Transfer learning.

Cite as: Discovering abstract concepts to aid cross-environment transfer for a learning agent, C.Herpson and V.Corruble, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra, and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX.

1. INTRODUCTION

Learning and transfer of knowledge is a cross-discipline issue for those interested in understanding or simulating intelligent behavior. It has been in particular addressed by research in cognitive psychology and neurosciences [7]. In Artificial Intelligence (AI), the ability to generalize has been an important focus of study, but mainly in the field of classification, i.e. for the identification of new instances of a given *concept* [16, 17]. Little effort has been put until recently into the transfer of learned knowledge to ease the accomplishment of new instances of a *task*. The main, significant exception to this is the field of Case-Based Reasoning (CBR) [14] which has provided a framework and tools for transfer with some considerable success. However, though CBR clearly considers transfer as a complete cycle, it hardly qualifies as an agent architecture as it is usually understood in the agent community. The perception/action loop is central to agent architectures whereas CBR usually considers information already modeled at a symbolic, abstract level (as evidenced with the key notion of *case*) more typical of traditional AI approaches. Also, as CBR's ambition is to provide a complete architecture (the loop is meant complete from the consideration of new cases to the decision/action step), it does not appear open to integrate easily alternate or additional mechanisms, be it for learning, reasoning, decision or other aspects of an agent architecture.

The need for an agent architecture which integrates learning and transfer capacities has become more crucial these last years with the development of some new application domains using, or open to the use of, autonomous agents, such as video games or military simulations. Work in the area of strategy games has lead to techniques which let agents learn strategies through playing [3]. Yet, learned strategies obtained with these techniques are only relevant to the game context in which they have been learned, that is to say a scenario, and more pointedly, a "game map" specific to this scenario. Hence, each new scenario requires a new phase of learning, which is usually time-consuming, since previous experience is not put to use.

The goal of this paper is to present a robust learning agent architecture with abstraction and generalization capacities which lets transfer knowledge learned on a given topology to a different one, yet unseen. In the following we will briefly introduce relevant literature and then describe the properties that we consider as necessary to reach an efficient knowledge transfer in this context. We describe next the proposed architecture, then the preliminary evaluation of this architecture in a simplified game environment. Finally, we discuss experimental results and current limitations of our approach before concluding with some perspectives.

2. Related work

We will distinguish two broad categories in the following. In the *General Game Playing* introduce by Pell [13], the task being studied is the ability of architecture to work efficiently in different environments by analyzing the rules that govern each ones and automatically build specifics representations for them. In this research field, the environments under consideration are finite, non-stochastic and of complete information. The other and more recent approach consider the transfer learning ability without the hypothesis of complete information. In both cases, a system, to reach high performances, should be flexible enough to adapt to variations within rules and/or environments.

In the last few years, both research axes on learning and transfer have found excellent testbeds in the area of games. Indeed, games provide research environments which are rich and complex, often stochastic, and favor experimental work [4].

In Real-Time Strategy (RTS) games, the complexity of the game leads to a high number of situations within a single game or across scenarios. A first approach of transfer models game states at a high level of abstraction meant to describe the state globally. At the opposite, another approach is to consider each unit or character of the game as an autonomous agent interacting with the environment and processing only the information it can perceive locally. Following this second approach, Gorski *et al* [2] tackles knowledge transfer within the SOAR cognitive architecture [8]. In SOAR, the agent knowledge is represented as rules. A number of learning mechanisms for the transfer of spatial knowledge have been tested in the Urban Combat game environment (a "first-person shooter"), for tasks of different complexity levels. These mechanisms are varied as they range from the application of the Rete algorithm [11], to the Soar algorithm to create new rules by chunking, to an approach using a Reinforcement Learning algorithm [9]. If the two first mechanisms rely on the storage of all spatial knowledge, the last one deal with a state-action space. The task considered in a first stage consists in having an agent that learns to reach a target location in its environment. The agent initial position and its target position can vary between learning and evaluation. In a second stage, an obstacle is placed after learning on the learned trajectory to force the agent to adapt its strategy.

Results showed that a simple RL algorithm, in contrast with the other methods tested, did not lead to good agent performances. Changing the map structure causes a major overhead in recomputing state-action values.

With the same application domain, Sharma *et al* [1] addresses the problem at a higher level of abstraction. It proposes an architecture where an agent controls from a central point of view all the units/characters of its side using a CBR learner, i.e. a combination of CBR and RL. Their proposal was tested on the MadRTS™ game simulator, on a scenario where each side aims at destroying enemy forces and controlling as much territory as possible.

At each time step, the system builds a high-level description of the game state using only global information such as the average health status of all units, percentage of units "alive" on the friendly side and on the enemy side, percentage of territory controlled, etc. This radical abstraction makes the representation independent from any given map or game context. The architecture then selects an action solely as a function of this description. Couples such as <abstract game description, selected action> are stored in the case base and a RL algorithm (TD(λ), [10]) is then used to reinforce cases which lead to good results. Since stored cases are location-independent, the knowledge learned can easily be transferred to similar problems, i.e. other game scenarios.

The results obtained with this approach indicate its ability to reuse learned knowledge when initial positions and/or number of units vary. However, the fact that the game state description on which decisions are made is completely unrelated to the context (including its topology) seems to constitute a major obstacle to more ambitious transfer. Thus, a map of higher complexity, or a complete change of environment will not impact the state description and lead therefore to only one high level description for two distinct situations. As a result, only one action will be chosen where two different actions have to be selected.

An alternative, following Gorski's approach implementing an agent-centered perspective on the problem, appears as it could potentially help the transfer of high-level knowledge. Nevertheless, learning techniques tested so far in this context do not seem flexible enough so as to permit an efficient cross-environment transfer learning. Indeed, storing all the topological information requires the use of huge amounts of memory with no latency access period. Furthermore, an only symbolic representation of all the information extracted from the environment will create difficulties to use acquired knowledge in new environments.

3. Requirements for operational Transfer-Learning

Considering that without the necessary resources an architecture will not be able to display this transfer ability, we constituted a list properties that we deem desirable to reach this goal and to obtain a robust architecture, even in stochastic environments.

Intuitively, and in a general point of view, an autonomous agent able to reuse knowledge acquired in various contexts must, at least, have available a memory, and also of a knowledge representation, as well as action-selection and learning mechanisms.

An architecture without learning mechanism can be designed (then the loading of all knowledge must be done at the agent's instantiation) but the resulting agent will not be able to adapt itself to new situations. In the way we consider it in this article, only learning offers adaptation ability (Darwinian mechanisms are not being considered).

These four functionalities can be seen as part of two distinct but interrelated set, storage and representation on one hand, manipulation and update on the other.

- *Memorization*: To store experience and be able to capitalize on it.
- *Dynamic context representation*: To offer a sufficiently high level of representation to be independent of the learning environment but sufficiently rich to allow a specialization on a specific problem.
- *Decision mechanism*: To manipulate learned knowledge and to confront it to observed facts in order to act efficiently, online.
- *Learning mechanism*: To recognize previously learned knowledge and adapt it, or to create new one, in order to improve action-selection in the current environment as well as in another.

Based on these guide-lines, the limitations of previously introduced work seem to be due to the learning mechanism and the representation choices. In order to go beyond these limitations, we focused on these properties in the architecture description that follows.

4. An architecture for cross-map transfers

4.1 Abstraction and situated representation

We consider a representation using the notion of a situated agent. It lets one change from a central, globalized point of view, often used in work for strategy games to an agent-centered perspective.

With this world representation, where "situations" as perceived by an agent in its environment are the basic level of information, the elimination of irrelevant detail, considered as a passive abstraction [6] and usually seen as a constraint in domains such as situated robotics [5], brings significant advantages here. Indeed, it offers a knowledge representation at a level high enough for the agent's reasoning to be independent of map-specific geographical locations (x,y coordinates) and of the environment's complexity, in terms of size as well as richness, of its environment. That is, according to previously highlighted properties, a key to an improved knowledge transfer.

However, let us recognize that by using agent-centered limited perceptions and representations, we lose some possibly relevant state information and enter the realm of partially observable environments. Furthermore, if the environment contains multiple agents, they become unobserved and unpredictable; which means that the environment is not stationary anymore. These properties, which are lost with our choice of representation, are usually considered needed in most learning agent frameworks, as they are necessary to guarantee convergence with typical learning algorithms. Doing without them requires special attention.

From this abstraction (centered and local, with limited horizon), we proceed with a change of representation to express a situation as a set of attribute-value couples. In the following, we will use "situation" to refer indiscriminately to what is perceived by the agent and to its description.

Relying on this representation, our architecture considers two degrees of abstraction, which we name respectively the tactical and the strategic level. While the tactical level manipulates elements of information and knowledge directly available to the agent from its sensors, the strategic level is dedicated to inferences of a higher level.

We focus in this paper on the description of the tactical level where the discovery and the use of new concepts takes place, key to the transfer capacity of our approach. The strategic level, which takes care of the aggregation of information coming from the tactical level, becomes key when addressing the issue of multi-agent cooperation, and is not addressed in this paper.

4.2 Architecture

Designed to fulfill the different properties previously introduced, our architecture is based on a perception-action loop involving several components.

As shown in Figure 1, an agent has a memory to save facts, learned concepts and rules about the environment (*Concept and rules' DataBases* in Figure 1). An inference engine (*Action Selection*) uses the relations between theses different elements in order to select the actions that can lead it to reach its current goal. The automatic recognition into the current situation of the relevance of a prior learned pattern (*Identification*) is realized in our architecture by a similarity measure. Finally, the learning mechanism (*Concept Learning*), using both supervised and unsupervised methods, extract, represent and associate to the game's rules (or other elements of the model of the environment) relevant concepts.

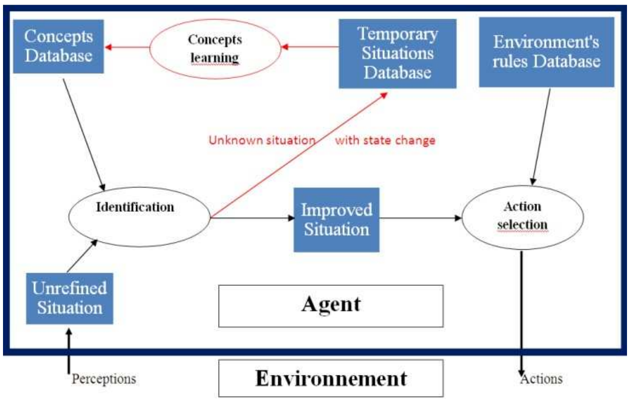

Figure 1: General architecture of a learning agent for cross-environment transfer.

Initially without knowledge, actions are in a first stage selected randomly at each time step. The agent then learns continuously to extract concept from situations perceived and to associate them to the premise of its decisions rules (red process in Figure 1) so that it helps reaching better decisions (decisions leading toward the goal) in future steps. Thus, concepts extracted from situations are used as an interface between the Physical sphere (the real environment), and the Representation sphere (knowledge manipulation).

Before presenting in some more details the mechanisms at work in the concept discovery phase as well as in their use in making decisions, we introduce an example that will be used to illustrate our subject.

Example : Suppose an agent, called *A*, participates in a fishing competition . The FisherAgent's goal is to get a fish before the end of the competition. Its internal state can vary from "*Fishing"*, "*Win"* to "*Empty-handed*". It is initialized to the "Fishing" state. By moving along the river bank, the agent can reach a submerged tree where a lot of fish hides, or remain in an open area of the bank. When the floater of its fishing line sinks, it can hook. If successful, it gets the fish and accomplishes its goal (internal state = "Winner"). If *A* misses the fish, its internal state ("Fishing") remains unchanged. If another competitor gets a fish first, *A* loses and its internal state switches to "Empty-handed". We therefore understand that the Hook action will depend on the observation of the fact "FloaterSink" which gives the FisherAgent a shot at victory.

4.2.1 Learning phase

The agent internal state is defined by one or more variables. An agent's change of internal state corresponds to a change in one of its attributes. A change occurs as a consequence of the agent's last action or of another agent's one.

Without prior knowledge, the learning or discovery of a new concept proceeds in three stages:

(1) Saving into the *Temporary Situation Base* all situation patterns newly encountered when a change in the agent's internal state occurs. Each saved situation is associated to a reference to the pre-condition part of the last action rule used.

(2) Apply a learning algorithm to infer a pattern when the number of situations associated with the same pre-condition is above a threshold N.

(3) Resulting pattern characterizes the situations encountered when executing the last action. The resulting couple : <pattern, pre-condition> is added to the *Concept Base*.

Example : In our example, the FisherAgent will go on a number of fishing outings recording each time the situation perceived when its internal state switches from « Fishing » to « Win ». After N outings, the agent learns that being located next to the submerged tree (to see it in its field of view) improves the chances to see the floater sink and then to win.

In stage (2), two types of learning algorithms are used depending on whether or not the target concept is related to the final goal. The reason is that we hypothesized that concepts linked to win or lose situations can be defined within a single representation. Assuming this is true, winning situations and losing situations can be opposed and form two classes which can be discriminated using a classical supervised learning algorithm (case A). However, if the target concept is a sub-goal, a process of characterization using an unsupervised learning algorithm is carried out (case B).

Example: In order to illustrate the notion of subgoal, we use a slightly more complex example: for the sake of fairness, the jury of the competition now requires all competitors to use the same type of bait. The variable used to describe the internal state of the agent, now with an additional value "waiting" in its domain, will switch to "Fishing" only when the subgoal "Presentation of baits to the jury" will have been reached. The only situations available to the agent to store will be the ones obtained when this subgoal is reached. This last point therefore requires the use of an unsupervised learning technique.

A) Concept related to the final goal, learned under supervision.

Environments in which agents evolve are, just as the real world, stochastic. The same final situation can therefore be associated sometimes to a win, sometimes to a loss, in two different episodes. There is therefore a significant amount of noise in the data from which we wish to learn the concept. To tackle this, the algorithm that we propose firstly deletes the noise from data then infers a pattern among remaining situations:

(1) Maximizing inter-class distance and minimizing intra-class distance by filtering out borderline situations. This disambiguation process is based on a majority vote. Intuitively, each situation in the base formulates hypothesis on the class of the other situations. The comparison between the majority's point of view and the effective class lead to put aside miss classified ones.

(2) Apply a supervised learning algorithm

Once these two stages have been realized, all the situations associated to the precondition of the action having triggered the learning phase are removed from the Temporary Situation Base. The situation base obtained through the disambiguation process and the resulting classification tree are added to the Concept Base, associated to the corresponding precondition in a triplet <precondition, disambiguated situation base, tree>.

B) Concept related to a sub-goal: process of characterization.

Characterizing the concept consists in identifying attributes whose values show a pattern within the subset of situations associated with one pre-condition. Let us consider that attribute L is representative of a given concept whenever more than α % of its instances lie within the interval defined by a variation of γ around its mean value. The resulting situation constitutes a prototype for the concept which originated the N situations under consideration. In a manner similar to what was presented in Case A above, situations used from the *Temporary Situation Base* are deleted, and the new concept is added to the *Concept Base*.

In order to distinguish the various processes at work in our architecture, we have so far considered that the *Concept Base* was initially empty during the recording of <new situation, precondition> couples in the *Temporary Situation Base*. However, the main goal of our architecture is to allow for the reuse of knowledge acquired previously. It needs to be able to use a concept already learned, and also to recognize the equivalence with a known concept, even if it has been labeled differently, for example: the same concept in another language. As a consequence, when the *Concept Base* is not empty during the recording of new situations in the *Temporary Situation Base*, a similarity measure is applied between new incoming situations and the descriptions of learned concepts. If a correspondence is detected (similarity above a threshold), the couple <pattern,precondition> in the *Concept Base* is updated with an additional precondition synonym and the incoming situation is deleted. The same similarity measure will be used in the decision phase presented in the following.

4.2.2 Decision phase

At each time step, the inference engine uses its model of the environment (possibly the game rules) and known facts to apply backward chaining so as to « prove » the desired goal. In case a missing precondition p blocks the action that would accomplish the goal, the inference engine stops and signals this difficulty by communicating the missing precondition. This information triggers the following process:

(1) The agent searches its Concept Base a concept associated to precondition *p*. If such a concept exists, it retrieves its description.

(2) If precondition *p* is related to a subgoal, the description obtained is the prototype of situations in which p was satisfied in previous experience. If p is related to the end goal; the agent has obtained an identification function.

(3) The agent then attempts to identify among the visible locations within its horizon those whose associated situation corresponds to the description extracted from the Concept Base.

(4) If no relevant situation has been identified, the agent chooses an action randomly among those available. If more than one situation corresponds to the target concept, the agent selects with probability δ the destination among these which is closest to its current location. To avoid behaviors such as going back and forth between two locations by agents always selecting the closest destination, they also select with probability η a destination selected randomly among those deemed relevant in step (3) above. Once a destination has been selected, the agent applies a search algorithm (A*) so as to select an immediate action leading toward the selected destination. Lastly, in order to insure a dose of exploration, even when a target situation has been selected, the agent selects a fully random action with probability ε (with $\delta + \eta + \varepsilon = 1$).

During the identification of stage (3), the agent considers virtually all the positions it can perceive and evaluates their quality relatively to the target concept. As we assume that, initially, the agent does not have available the map of the environment, its only source of information regarding virtual positions is the map of the environment that it builds gradually during its moves. The identification carried out at each location works in two different manners depending on whether the description obtained from the Concept Base is an identification function (Case A: concept related to the end goal) or a prototype (Case B: concept related to a sub-goal).

A) *Identification of a concept related to the end goal*.

The identification function filters in a first stage the virtual situations detected within the horizon of the agent, using the disambiguation process previously introduced. The remaining situations are given as inputs to the classification tree. This one computes for each situation its probability to be relevant to the concept the agent is looking for. The tree computes a value in the [0, 1] interval, where 1 means full probability and 0 the opposite. To limit the risk of errors resulting from misclassification, only the virtual situation with a confidence value above a threshold β (β in [0,1]) will be effectively selected for the following.

B) *Identification of a concept related to a sub-goal.*

In this case, the element extracted from the Concept Base is a situation prototypical of the target concept. A non-informed similarity measure is applied (only considering information contained in the situations considered). We use the notion of distance as presented in [15] to estimate similarity over attributes describing the various situations. The choice of a specific similarity measure strongly impacts the quality of the results. The Euclidian distances was so selected after initial experiments with Manhattan, Euclidian and Mahalanobis distances.

5. Experimental evaluation

5.1 Experimental setting

We chose a grid-world type of environment to test the agent architecture presented above. Each location on the grid is characterized by its altitude, besides its (x,y) coordinates.

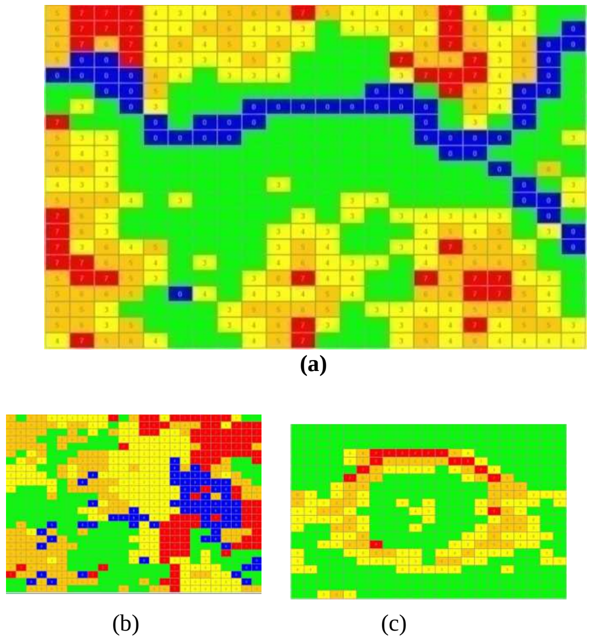

Figure 2: Maps used to evaluate the transfer of knowledge. Altitude varies from blue (level 0, the lowest) to red (highest). Figures (a) and (c) respectively correspond to the Mountain and Corrie test environments; the figure (b) is the learning one.

In order to evaluate the learning and transfer capacity of our architecture, we set up 3 maps with varying topologies, sizes, and maximum altitudes. To insure that agent locations would impact the situations it could perceive, the effective field of view of the agents (covering a 180° angle) takes into account the obstacles present (typically a hill can block the view). As a consequence, the field of view of an agent located on a high position is better than the one of an agent located downhill.

Range of sight	Current lowest altitude (in Fov)
Current orientation of A	Current highest altitude (in Fov)
Last action done by A	Current Mean altitude (in Fov)
Highest altitude seen by A on this map	Homogeneity
Current agent altitude (relatively to highest altitude)	Components in A's Fov
FieldOfView area	Components in A's square

Figure 3 : A situation is the abstraction of the environment realized by the agent. It is here reformulated as a set of attribute/value couples. All elements except the two last ones are numeric.

The chosen scenario is somewhat similar to a predator/prey simulation in the above grid world. More precisely, it consists in a duel between two agents evolving within the environment. Each agent's goal is to become the last survivor. They have available a range weapon and have to hit their competitor twice to win. Each action's availability (move and rotation towards the 4 directions, i.e. 8 actions) depends on the agent's current situation. The shoot action is automatically executed when an agent perceives its enemy in its field of view (ammo(x) & see(x,y)→Shoot(x,y)). This was decided so as to reduce the simulation time. The result of the shoot action (target hit or not) is stochastic and depends on a number of characteristics unknown to the agent. The probability of hitting when shooting depends on the distance shooter-target, as well as the angle of incidence of the shooting: an horizontal shoot has a very high hit probability, whereas when 2 agents are located at different altitudes, the one positioned higher will have a hit probability much higher than the lower one.

5.2 Experimental setup

In order to test the impact of learning a single concept on the performances, changes of internal states leading to the storing of situations and the learning of a concept will only be related to end-game situations. We will therefore consider that ammunitions are unlimited. The learning algorithm used in our case is a probabilistic classification tree based on the C4.5 algorithm [12]. It uses one third of the elements saved after the disambiguation process (4.2.1 - A) for learning and the two third remaining to auto evaluate the quality of the learned tree.[1]

The β parameter defining the tolerated dissimilarity between two situations referring to the same concept is set to 0.8. Parameters α and γ defining what constitutes a relevant attribute for a given concept are set respectively to 0.75 (overqualified majority) and 0.1. Parameter δ, defining the probability to select the closest destination under the possible ones is set at 0,8 and parameters η and ε are both set at 0,1.

Performances are evaluated based on 3 criteria:

- The percentage of victories obtained by each agent
- The average number of time steps needed to end an episode or game (i.e. when a agent wins)
- The evolution of the proportion of the environment explored by each agent over time.

The first experiment aims at evaluating the ability of an agent with one learned concept (related to the end goal) at transferring knowledge to a new environment. We run 2000 episodes placing 2 random agents (they select their action randomly among them available in their current situation) on the learning environment. Situations associated to internal state changes are stored but the learning step is not activated. Then we run the concept learning algorithm for one of the two agents. Third, we deactivate the learning again and we run series of 1000 episodes opposing a random agent and a trained agent on various test environments different than the one used for training. To obtain a baseline, 1000 duels between two random agents are also run for each test environment considered.

The second experiment aims at evaluating the capacity of our architecture to produce a behavior of a better efficiency by adding an additional memory component. Indeed, in the first experiments, stimuli from the agent's sensor do not provide any temporal information. An agent is not able to know if it is currently in an area it just left a few steps ago. Consequently, it can move for a long time in an area where there is no enemy. If information from the new component, which is added to the description of each situation, is recognized as relevant during the concept learning, it may improve the results because of a better spatial exploration.

Following the same method, an agent thus « upgraded » is placed in the training environment against a random agent. After 1000 episodes, the learning mechanism is activated and then the agent with the new learned concept is placed in the various test environments where it is tested over 1000 duels.

In the presentation of all results, we refer to the 3 types of agents tested as follows:

- *Random* agent: agent with no knowledge nor learning mechanism
- *Intel_1* agent: agent having discovered/learned a concept related to the end goal.
- *Intel_2* agent: agent with a short-term memory having learned a concept related to the end goal.

5.3 Results

The tree learned in the first experiment shows (see Figure 5 below) that the agent will favor situations with a high altitude and a good field of view, or, if the field of view is considered average, situations where average altitude around the agent location is lower than its own.

[1] The classification tree was selected after having tested a number of learning algorithms including Clustering, KNN, Kmoy, Boosting on Multi-Layer Perceptrons and Support Vector Machines. Though the accuracy obtained with the selected method is slightly lower than the one obtained with an SVM, it is better in the sense that it is easily interpretable by humans, which lets one control which concepts are considered significant by the agent.

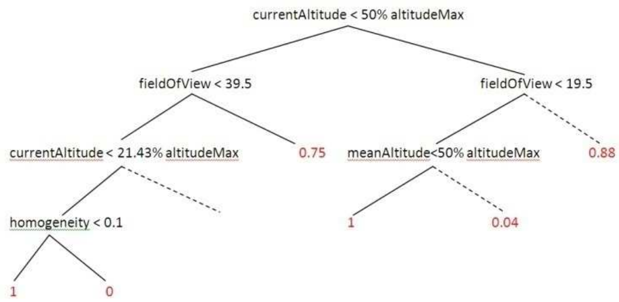

Figure 4: Tree automatically generated and associated to the precondition « see(x,y) » for the « shoot » action. A value above 0.75 indicates a favorable situation. Dotted lines indicate subtrees that were manually pruned from the figure for better clarity.

The results obtained for the first evaluation criterion on the three environments are presented in Figure 6. It shows that the concept learned is sufficiently relevant to improve performance on the training environment. Furthermore, the results obtained on the two test environments are even better than the one from the training environment.

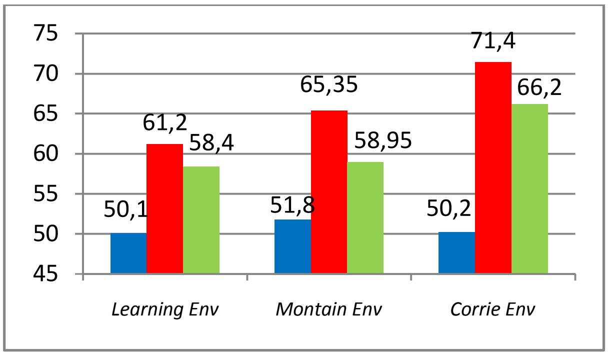

Figure 5: % of wins for respectively agents random, Intel_1 and Intel_2, all against a random agent, in 3 different environments after 1000 duels.

Agent *Intel_1* improves results over a random agent by 22% in the training environment and by 26 to 42% on test environments. Agent *Intel_2* improves results by 15 to 32% depending on the environment. However it seems that its memory is more of a hindrance here if compared to the results of *Intel_1*.

Results obtained in Figure 7 with the second evaluation criterion (the length of a duel) explained this last result better: *Intel_1* increases the average duel time for 2 of the 3 environments while *Intel_2* drastically reduces the time on all three of them.

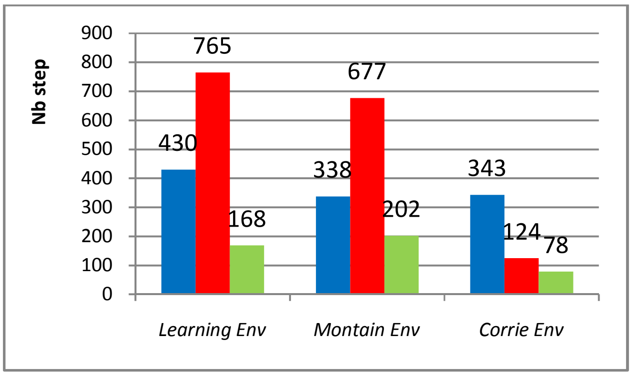

Figure 6 : Average duration of a duel for random, *Intel_1* et *Intel_2* agents facing a random agent in 3 environments, after 1000 duels.

Results of *Intel_2* indicate that the temporal information is profitable.

For the third evaluation criterion, the general shape of graphs is similar for all 3 environments so we give below in Figure 8 the results for the Mountain environment only.

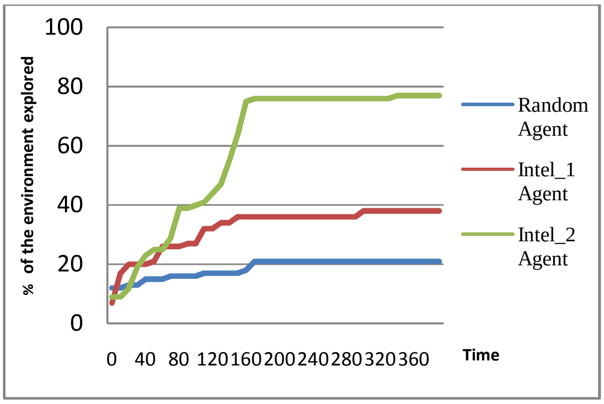

Figure 7 : Evolution over time of the proportion of the environment explored, for each agent in the Mountain environment.

Because of its memory the *Intel_2* agent goes over a larger part of the environment in a given time interval. It is therefore more likely to encounter its opponent than *Intel_1*. This last one, without short-term memory, cross repeated times the same zone. In this context, *Intel_1* has an advantage as actions are simultaneous, the one moving can very well enter the field of view of its opponent and get shot immediately. Agent *Intel_1* takes fewer risks, and therefore plays longer and wins more often.

6. Discussion

Though our preliminary results have demonstrated the validity of the principles behind our learning architecture for the transfer of knowledge, a lot of work remains to prove its relevance outside the simple experimental setting used here. A first step in this direction would be to evaluate the architecture on a more complex environment to check its robustness when scaling up. Besides this point, our architecture suffers in its current form of several limitations.

First, the concept learning mechanism is triggered only when the number of situations available in the *Temporary Situation Base* is higher than the set threshold (N). This leads to a discontinuous behavior for the agent. A learning grounded on a progressive modification of the concept description could allow an agent to adapt more quickly to new environments and new associated concepts. This kind of approach would require an incremental approach in order to modify the impact of a new situation on the description of an already existing concept relatively to its age.

Another limitation of our architecture in its current form lies in the inference mechanism that it uses. It evaluates the satisfaction of the precondition of a rule by testing them from the left to the right. Since the satisfaction of some conditions directly depends on the dynamic situation of the agent, the inference engine will induce an oscillatory behavior. With the rule *Ammo(x) & See(x,y)*→*Shoot(x,y)* expressed in this order, the agent will secure ammunitions before it looks for its opponent. If preconditions are expressed in a reversed order, the agent will enter an endless loop "looking for the enemy–looking for ammunition". An algorithm learning which automatically order the preconditions should be learned would be useful here.

Furthermore, as witnessed by the difference in victory levels between *Intel_1* and *Intel_2* against a random agent, our architecture lets our agents identify desirable situations within their environment, by making use of the rules of the game. They are however not able to reason on what constitutes undesirable situations for them. For example, agent *Intel_2* learns to move so as to be in a good position to see the Random agent but does not take into consideration that moving is dangerous in itself. Using the game rules (or more generally a model of the environment) to model the goals of the opponent to avoid situations which would be desirable to him/her would significantly improve the performances of our architecture. With this approach, we can even hypothesize that *Intel_2* would overcome *Intel_1*.

Besides these limitations, and from a higher perspective, our approach bears some similarity with, but is also clearly different in its principle, to the one proposed in [1]. The RL algorithm is a key feature of their architecture. At the opposite, ours lets an agent learn to choose a state with the goal of executing an action resulting from a reasoning process. In that sense, ours is more cognitive in nature

7. Conclusion and Perspectives

We have proposed a new learning agent architecture for the transfer of knowledge respecting four properties that we consider as necessary to reach this goal: Memorization, Context free representation, Decision process and Learning process.

Initial results are encouraging. They show that our architecture does lead to a relevant knowledge transfer in stochastic environments regarding topological information, which proves efficient when the environment changes. Besides some necessary improvements discussed in section 5, one short-term perspective is to evaluate the efficiency of our learning architecture facing the Reinforcement Learning approach. This experiment will permit to compare both adaptability and efficiency of our approach to the RL ones in the transfer learning context.

With a long-term view, an important perspective for this work is to extend the architecture to address a multi-agent framework, focusing on the strategic level of representation which will be key to the issue of coordination. The choice of knowledge representation proposed here is compatible with the use of a command hierarchy where low-level units with local information can interact and share their knowledge at the strategic level. We aim at tackling the issue of communication and coordination between agents at this level.

8. REFERENCES

[1] M. Sharma, M. Holmes, J.C Santamaria, A. Irani, C. Lee Isbell Jr., A. Ram: Transfer Learning in Real-Time Strategy Games Using Hybrid CBR/RL. In *International Joint Conference on Artificial Intelligence*, 2007: 1041-1046.

[2] N. Gorski and J. Laird. Experiments in transfer across multiple learning mechanisms. In *ICML Workshop on Structural Knowledge Transfer for Machine Learning*, 2006.

[3] Madeira, C.: Adaptive Agents for Modern Strategy Games: an Approach Based on Reinforcement Learning, Ph.D. Thesis, University of Paris VI, 2007.

[4] D. Aha, M. Molineaux, and M. Ponsen : Learning to win: Case-based plan selection in a real-time strategy game. In *ICCBR*, pages 5–20, 2005.

[5] N. Bredeche, Z. Shi, J-D. Zucker. Perceptual Learning and Abstraction in Machine Learning : an application to autonomous robotics. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Vol. 36, Issue 2, pp.172-181, march 2006.

[6] Agre. P. E, 1988 : The dynamic structure of everyday life. *PHD thesis*, Cambridge (MIT) artificial Intelligence Lab

[7] S. Thrun , L. Pratt, Learning to learn, Kluwer Academic Publishers, Norwell, MA, pages 45-71, 1998.

[8] Laird, Newell, Rosenbloom : Soar : An architecture for general intelligence, Artificial Intelligence, 33(1) : 1-64, 1987.

[9] Sutton, R. Sutton & Barto, A.G. Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA,1998.

[10] DAYAN P., SEJNOWSKI T. J., « TD(λ) Converges with Probability 1 », Machine Learning, vol. 14, n°3, p. 295–301, 1994.

[11] C L. Forgy : Rete : a fast algorithm for the many pattern / many object pattern match problem, IEEE 324-341, Los Alamitos, 1990.

[12] Breiman, L., et al., Classification and Regression Trees, Chapman & Hall, Boca Raton, 1993.

[13] Barney Darryl Pell : Strategy Generation and Evaluation for Meta-Game Playing .Ph.D thesis, University of Cambridge, 1993.

[14] Watson, I. and Marir, F. (1994). Case-based reasoning: a review. *The Knowledge Engineering : Review*, 9(4):327–354.

[15] Gilles Bisson : La similarité : Une notion symbolique / numérique, Induction symbolique numérique à partir de données, p169-223, 2000.

[16] Steve R. Gunn : Support Vector Machines for Classification and Regression, ISIS Technical Report, University of Southampton, 1998.

[17] Christopher M. Bishop : Pattern Recognition and Machine Learning, Springer, 2006.