# Combining Reinforcement Learning with a Multi-Level Abstraction Method to Design a Powerful Game AI

Charles Madeira Vincent Corruble

LIP6, Université Pierre et Marie Curie, 4 Place Jussieu, 75252 Paris Cedex 05, France

## Abstract

This paper investigates the design of a challenging Game AI for a modern strategy game, which can be seen as a large-scale multiagent simulation of an historical military confrontation. As an alternative to the typical script-based approach used in industry, we test an approach where military units and leaders, organized in a hierarchy, learn to improve their collective behavior through playing repeated games. In order to allow the application of a reinforcement learning framework at each level of this complex hierarchical decision-making structure, we propose an abstraction mechanism that adapts semi-automatically the level of detail of the state and action representations to the level of the agent. We also study specifically various reward signals as well as inter-agent communication setups and show their impact on the Game AI performance, distinctively in offensive and defensive modes. The resulting Game AI achieves very good performance when compared with the existing commercial script-based solution.

**Keywords**: reinforcement learning, strategic decision-making, modern strategy games, abstraction, terrain analysis, multiagent systems.

**Authors' contact**:
charlesandrye@gmail.com
vincent.corruble@lip6.fr

## 1. Introduction

Modern strategy games constitute large-scale multiagent simulations that share many characteristics with real-world problems. Specifically, they reach high levels of complexity in terms of the sizes of state and action spaces, and their environments are partially observable and stochastic. For these games, it is generally recognized that the design of challenging automated opponents (called *game AI*) is an important issue [Buro 2003]. The most common approach to this problem in the game industry is to use ad-hoc techniques with hand-crafted knowledge, although this approach is tedious and usually leads to low-quality opponents [Nareyek 2004; Rabin 2003; Rabin 2006].

Let us consider, as a motivating example, the modern strategy game shown in the screenshot of Figure 1. This is Battleground™ (Talonsoft®), a commercial wargame which simulates historical Napoleonic battles (see http://johntillersoftware.com/NapoleonicBattles.html), therefore focusing on the military aspects (rather than economic, diplomatic aspects treated extensively in other strategy games).


Figure 1: Battleground™ is a turn-based stochastic game that simulates the confrontation between two armies at historical battlefields.

Because of its parallel nature (at each turn, *all* military units can act simultaneously), the complexity of action selection for each side, seen from a centralized perspective, grows exponentially with the number of units it controls (if there are $U$ units, each with $A$ possible actions, the resulting branching factor is $A^U$). This property is well known in the multiagent systems literature, including in video-game applications [Guestrin et al. 2002; Guestrin et al. 2003]. Regarding the configuration of a simple Battleground™ scenario, the two armies contain respectively 101 and 83 front-line units that are placed on a small map of 35x20 hexagons. The combinatory explosion leads to a huge state space in the order of $10^{2000}$ and a maneuver action space of $10^{180}$ concerning one army and of $10^{150}$ concerning the other. As a result, a distributed approach to the design of a Game AI for this type of game is a natural and reasonable candidate to be explored. However, distributing action selection to the level of individual units (which are numerous – in the hundreds, heterogeneous – infantry, cavalry, artillery with dynamic size and health, and evolve in a complex environment – thousands of hexagons each characterized by a type of terrain, altitude, etc.) leads to another problem well studied in the multiagent community: how to coordinate the individual choices from all units. So that, though taken in a distributed and autonomous manner, individual decisions lead to a meaningful collective behavior.

The coordination problem is studied from many angles in the multiagent community, and we refer here especially to the classification proposed in [Claus and Boutilier 1998], which introduces learning as one approach to obtain or improve coordination between agents in a multiagent system. Since a game environment provides a straightforward (initially) reward signal – the game score, within game or at end-game situations, as well as the ability to repeat a number of learning episodes relatively easily, we have chosen to study the use of reinforcement learning (RL) [Sutton and Barto 1998] on this task.

RL is specifically designed for learning behavioral strategies by maximizing a cumulative reward from the interaction with the environment. A natural question is to ask whether state-of-the-art RL techniques are up to the job of tackling the complexity of learning for the type of games we are interested in. Results obtained by innovative systems such as TD-Gammon [Tesauro 2002] would seem encouraging as they have pushed back the limitations of the approach. However, it turns out, as illustrated by the Battleground™ simple scenario given above, that the problem we want to tackle dwarfs in complexity the ones typically addressed by the RL community.

Taking some inspiration from human intelligence, we describe in this paper an abstraction method that takes into account a hierarchical decision-making structure to design proper and tractable state and action representations for RL in a largely automated fashion. We integrate this new abstraction method with state-of-the-art RL techniques to construct a system that is able to design automatically efficient strategies for modern strategy games.

We apply our approach to the Battleground™ game and compare experimentally various setups of our learning agents with baselines techniques (script-based agent and random agent) and a human player. In order to explain these evaluations, we address briefly other challenging issues which had to be tackled, such as the design of a learning scenario, communication and coordination within the multiagent system, the design of adequate reward functions, and generalization through function approximation.

The results show that our system fairs very well against the existing commercial script-based solution. We conclude by outlining possible improvements and directions for future work.

## 2. Background

A promising approach for reducing complexity in learning problems consists in simplifying the level of detail of the information available [Giunchiglia and Walsh 1992; Blum and Langley 1997; Saitta and Zucker 2001]. There are several possibilities to change from a representation with full information to one at an abstracted level. One can, for example, remove irrelevant variables, generalize information, add constraints, etc. [Jong and Stone 2005; Li et al. 2006].

Knowledge about the domain such as military decision-making and spatial information is of crucial importance in the case of modern strategy games because it can guide us toward the purpose of abstraction [Corruble et al. 2002]. In this context, relevant information can be obtained by techniques known as terrain analysis.

Terrain analysis is a process that provides guidelines for gathering, analysis, and organization of intelligence, identifying areas of the battlefield that affect courses of action [Grindle et al. 2004]. It interprets natural and man-made features of a geographic area to determine their effects on military strategic and tactical maneuvers. Strategic maneuver is considered to be high-level decision-making where leaders can obtain a broad overview of the battlefield. Tactical analysis provides the leader with a much more detailed view of specific areas-of-interest on the battlefield.

Understanding terrain is especially useful because we can identify ideal locations for scouting parties, best line of sight/fire and also the ability to hide troops and equipment. These are hints that can be employed in the decision-making system indicating tactical suitability of the various locations for both attacks and defenses. The importance of the study and analysis of terrain has been recognized for hundreds of years in military science. Moreover, its importance has been recently recognized in the domain of modern strategy games [Forbus et al. 2001; Rabin 2006].

## 3. Abstraction Method Based on Terrain Analysis

In this section, we describe a new abstraction method based on terrain analysis that makes use of a hierarchical decision-making structure to adapt semi-automatically the level of detail of the state and action representations to the hierarchical level of the decision being made. Differently to standard hierarchical RL approaches such as MaxQ or HexQ [Dietterich 2000], our approach does not decompose agent tasks into subtasks. Instead of this, each agent always perform a unique kind of task by treating state and action spaces in an appropriate level of granularity.

### 3.1 Hierarchical Structure of the Decision-Making

In modern strategy games as indeed in many real-life situations, one can see that the military hierarchical structure of command and control is a natural candidate for the decomposition of the decision-making process. In this structure, terminal nodes correspond to units placed on the terrain and non-

terminal nodes represent the leaders of the groups of units (see Figure 2). The ability for decision-making is distributed across specific groups. Subordinate leaders receive orders from their superiors and use their domain-specific knowledge and local information to make their own decisions.

A main advantage of this structure is that it allows high-level leaders (who do not act directly on the environment because they cannot carry out physical actions) to make strategic decisions to accomplish high-level goals, only considering information at an appropriate level of detail (in this case, little detail but wide scope). It reduces the complexity of learning for each group of agents. However, the number of low-level units to be controlled and the intensity of interaction between their strategies increase dramatically when learning strategies for deeper levels of the hierarchy. This requires some higher degree of coordination between the leaders in order to provide a mechanism in which the units work as a team [Claus and Boutilier 1998; Guestrin et al. 2002].

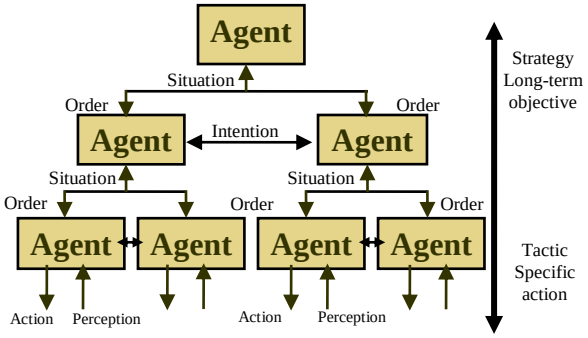


Figure 2: Hierarchical decomposition of an army in specialized groups of agents.

In this context, hierarchies can provide coordination mechanisms by modeling communication between agents as follows: (1) vertically, between a high-level agent and its subordinates (goal sharing); and (2) horizontally, between agents of the same group (intra-group communication). The challenge then consists in establishing a trade-off between the individual goal taken by each agent and the goal of its group.

## 3.2 Abstraction of State and Action Spaces

Strategic decision-making at the high levels of the hierarchy requires adequate state and action representations. For this purpose, we take some inspiration from terrain analysis techniques to propose a new algorithm for the abstraction of state and action spaces that integrates terrain data together with dynamic battlefield information. The high-level environment representation generated is used by leaders in order to get only the information they need to make relevant decisions. The abstraction process is executed in two main steps: abstraction of the action space and abstraction of the state space.

### 3.2.1 Abstraction of the Action Space

Tactical action spaces (lower-level actions used for tactical decisions) of modern strategy games are often composed of the combination of two main variables: a location on the map and an elementary task to accomplish at this location. In order to adapt tactical action spaces at the strategic level, we abstract both variables. On one hand, we gather manually a certain number of elementary tasks (fire, turn, move, etc.) to define high-level tasks (monitoring, exploration, occupation, concentrated attack, etc.).

On the other hand, our algorithm explores automatically the map by a rule-based approach in order to find key locations that reduce considerably the size of the action space. A *key location* is any position on the map whose control is likely to give distinct military advantage to the force that holds it. It could be a piece of high ground with good observation and fields of fire, transportation choke-points such as a water crossing, mountain gap, or road junction, or even dense woods or rivers that anchor the flank of a conflict line. It can be controlled by fire, obstacles, or the relative positioning of friendly forces and is often selected for conflict positions or objectives.

Finding key locations is then fundamental since it can let the leader to make inferences about possible degree of vulnerability of friendly forces to enemy attacks. Figure 3 illustrates the identification of key locations on a real map to demonstrate the generality of our algorithm. As a result, once high-level tasks are given and key locations are identified, they are combined to build the strategic action space.

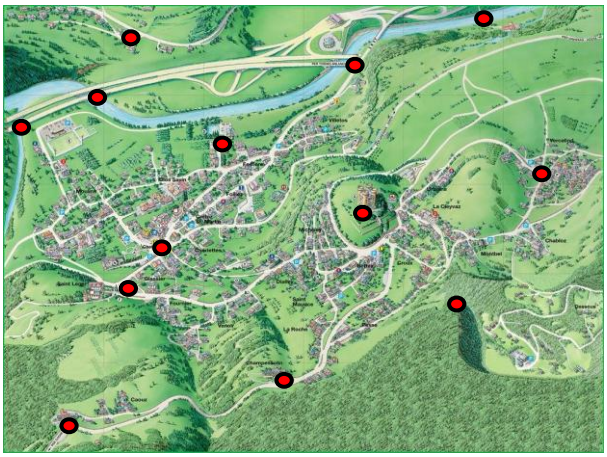

Figure 3: Identification of key locations on a real map. Bridges on the river, the castle on the top of the hill, the top of the mountain, large crossroads in the center town, crossroads of access for the city, are all good examples of key locations.

### 3.2.2 Abstraction of the State Space

State spaces of modern strategy games are composed of the combination of a number of variables which are used to represent full game situations. Our proposal for

abstracting them consists, on one hand, in generating a summary of the state of all units belonging to a group in the hierarchy. This summary can be composed of relevant variables such as center of mass, size, health, quality, mobility and ammunition of the group.

On the other hand, the key locations identified previously in the abstraction of action space are used as seeds to be expanded progressively by adding neighboring locations until a full partitioning of the map into strategic zones has been reached. A *strategic zone* is a region of the map where intelligence collection efforts must be focused. It is associated to a category of mobility (easy mobility, restricted mobility or no mobility) according to the terrain nature. Figure 4 illustrates strategic zones generated using as seeds the key locations identified on the real map of Figure 3.

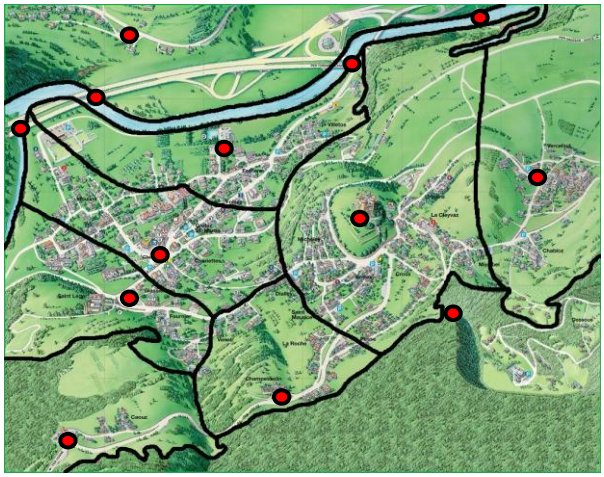

Figure 4: Identification of key locations and strategic zones on a real map. The strategic zones are generated by propagating key locations in order to aggregate adjacent locations, taking into account physical constraints of the terrain.

The state representation is composed of two main datasets: a summarized description of a group of units for which an order is under consideration, and a summarized description of friendly and enemy forces for each strategic zone on the map.

### 3.3 Algorithm Definition

Our algorithm (see Figure 5) takes into account some features of a given scenario: objectives, fixed objects (ease of access and obstacles), and a two-dimensional map topography. In addition to this, it takes also as input a set of players, a set of high-level tasks, a set of variables describing the situation of a group, and a set of variables describing the situation of units placed on strategic zones. The algorithm is composed of four steps as described below:

- **Identification of key locations**: locations, fixed objects and objectives on the map are evaluated by the ***isRelevantLocation*** function to identify those which are likely to give advantage to front-line units. Two factors can render key locations: how a unit wants to use it, and whether his enemy can use it to defeat the unit. So, a position is identified as key location when it is recognized that the task depends upon its seizure or retention. Examples include urban areas, lines of communication and supply, topography, drainage characteristics, bridges, choke points, high ground, key military installations, and supply routes.
- **Identification of strategic zones**: this step is divided into three sub-steps. First of all, locations whose associated terrain is of restricted mobility are identified on the map by the ***isTerrainOfRestrictedMobility*** function in order to generate preliminary strategic zones. Each non-aggregated location of this terrain type is used as seeds to create a new strategic zone and is then expanded progressively by adding adjacent locations of the same terrain type using the ***propZoneSameTerrain*** propagation recursive function. After this, the key locations identified in the previous step are also used as seeds which are expanded progressively by adding non-aggregated adjacent locations using the ***propZones*** function until a full partitioning of the map into strategic zones has been reached. The expansion rate of each location takes into account the relation between two factors: a minimum threshold and an accumulated weight of aggregation. The minimum threshold is a parameter that must be achieved for allowing the addition of an adjacent location to a zone. It is computed taking into account several criteria: the movement cost of the terrain type, the eventual obstacles and ease of access, as well as the variation of altitude. The accumulated weight of aggregation is a parameter that indicates the probability for adding a location to a zone. It is initialized to zero and it is reinforced after each aggregation failure. Finally, eventual "lost" strategic zones are created if there still are locations non-aggregated on the map. The procedure stops when there is no more free locations on the map. All strategic zones created are then combined in a single list;
- **Fusion of strategic zones**: strategic zones are evaluated by the ***fusionZones*** function to detect those in which size is relevant when compared to the map size. Small zones are absorbed by adjacent zones of larger boundary;
- **Generation of state and action spaces**: a state representation is built based on the results of the three first steps, by a projection onto a predefined list of relevant variables describing the situation of a group and the

combination of three sets (strategic zones, variables describing the situation of units placed on each zone, and sides). An action representation is built based on the Cartesian product of high-level maneuvers and key locations.

```
1:genStateActionRepresentation(Objectives,Objects,MapLocations,
2:   Players, HighLevelTasks, GroupDescription, ZoneDescription)
3: {
4:   KeyLocations ← ∅
5:   StrategicZones ← ∅
6:   for all o_i ∈ (Objectives ∪ Objects ∪ MapLocations) do
7:     if isRelevantLocation(o_i) then
8:       KeyLocations ← KeyLocations ∪ {o_i}
9:       StrategicZones ← StrategicZones ∪ {{o_i}}
10:    end if
11:  end for
12:  RestrictedZones ← ∅
13:  for all t_i ∈ MapLocations do
14:    if isTerrainOfRestrictedMobility(t_i) then
15:      Zone ← {t_i}
16:      Zone ← propZoneSameTerrain( Zone, MapLocations )
17:      RestrictedZones ← RestrictedZones ∪ {Zone}
18:    end if
19:  end for
20:  StrategicZones ← propZones(StrategicZones, MapLocations)
21:  LostZones ← ∅
22:  for all t_i ∈ MapLocations do
23:    Zone ← {t_i}
24:    Zone ← propZoneSameTerrain( Zone, MapLocations )
25:    LostZones ← LostZones ∪ {Zone}
26:  end for
27:  StrategicZones←StrategicZones ∪ RestrictedZones ∪LostZones
28:  StrategicZones ← fusionZones( StrategicZones, MapLocations)
29:  ActionRepresentation ← HighLevelTasks x KeyLocations
30:  ZoneRepresentation←ZoneDescription x StrategicZones x Players
31:  StateRepresentation ←GroupDescription ∪ ZoneRepresentation
32:  return ActionRepresentation and StateRepresentation
33: }
```

Figure 5: Sketch of algorithm for terrain analysis and abstraction of state and action spaces.

### 3.4 Generating Representations for Battleground™

In this section, we apply our abstraction method to Battleground™. For this purpose, we use the game scenario called *Death in the Flèches*, discussed previously in the introduction. It simulates a stage of the battle of Borodino (1812) between the French army of Napoleon and the Russian army of Kutuzov.

To design a strategic action space, we choose five high-level tasks predefined for the commercial game:

- **Extreme attack**: units attack in a more concentrated formation, with maximum stacking in each location;
- **Regular attack**: units move so as to take a specified location;
- **Wait order**: units stop movement;
- **Regular defend**: units move so as to hold a specified location;
- **Extreme defend**: units do not fall back until all of the units of the organization are no longer in good order.

To design a strategic state space, we choose 8 variables to describe the situation of groups and 2 variables to describe the situation of units on each zone (see Figure 6).

**Group description (8 variables)**

- Center of mass (x, y) of the group on the map (2 variables);
- Artillery, cavalry, and infantry strength levels (3 variables);
- Fatigue, quality, and movement allowance levels (3 variables).

**Strategic zone description (2 variables)**

- Unit strength and fatigue levels (2 variables)

Figure 6: Variables chosen to describe the state of groups of units and zones.

The execution of our algorithm of terrain analysis obtains as result 8 key locations and 6 strategic zones on the map (see Figure 7). So, a state representation composed of 32 variables arranged in two main datasets is built for the higher level of the hierarchy: 8 variables for group description, and 24 variables for strategic zone description (2 description variables x 6 strategic zones x 2 sides). In addition to this, high-level tasks are associated with key locations on the map to compose an abstracted action space of 33 strategic actions (4 high-level tasks x 8 key locations + *Wait order*).

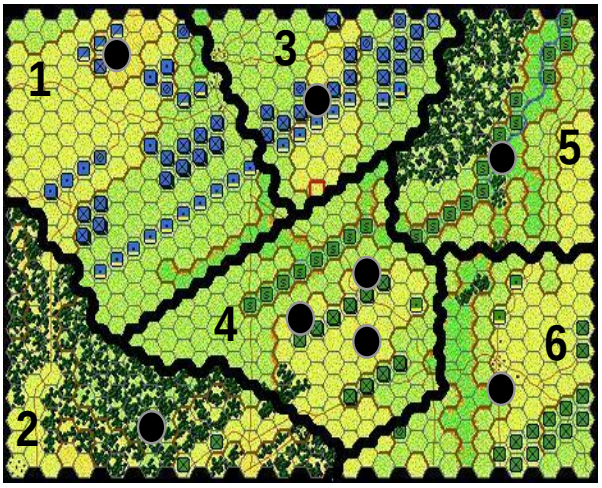


Figure 7: Our algorithm of terrain analysis identifies 8 key locations and 6 strategic zones on the map.

Consequently, the complexity of the scenario is considerably reduced to a state space in the order of $10^{82}$ and an action space composed of 33 actions for each leader. This is a significant achievement when compared to the original problem complexity presented in the introduction (state and action spaces in the order of $10^{2000}$ and $10^{180}$ respectively). Yet the real value of this achievement can only be measured on the basis of the performance of the whole system. This is what we aim for in the experimental evaluation presented in the next section.

# 4. Experiments

In this section, we present experiments using our approach applied to Battleground™ to evaluate the performance of the whole system. For this purpose, we address in the next subsection a number of crucial technical issues in order to obtain a fully functional system.

## 4.1 Learning Setup

In order to speed up learning, we set up our system with a particular learning scenario, a function approximator for generalization and a communication method to favor coordination among agents. These points are discussed below before coming to the experimental results themselves.

### 4.1.1 Learning Scenario

The learning scenario is configured according to a particular bootstrap mechanism proposed by Madeira [2004] that is able to accelerate the reinforcement learning procedure. This mechanism generates valuable training data as it leaves only a small part of the hierarchical decision-making to be learned at any given time. In effect, we configure our learning system (called learning AI) to control and learning only the decision-making at the higher level of the French army, giving strategic orders to its subordinates. The script-based decision-making system included in the Battleground™ game (called bootstrap AI) is used to control the Russian army, and the lower levels of the French army according to the orders flowing down from the level controlled by the learning AI. We also test in a second stage the situation where roles are reversed, control and learning a Russian strategy against the French. This is relevant as the battle is not symmetrical, since the French army is attacking (his goal is to conquer some locations controlled by the Russian army on the start of the game) and the Russian one is defending (his goal is to defend some locations from the French army attack).

### 4.1.2 Non-linear Function Approximation

In order to generalize between similar situations, a three-layer artificial neural network trained by the error back-propagation algorithm is employed. It is composed of 32 inputs $v_1,v_2,\ldots,v_{32}$ corresponding to the variables of the state representation $s=(v_1,v_2,\ldots,v_{32})$, and 33 outputs $Q(s,a_1),Q(s,a_2),\ldots,Q(s,a_{33})$ corresponding to the cumulative rewards associated with the action space $A=\{a_1,a_2,\ldots,a_{33}\}$. Cumulative rewards of selected actions are updated incrementally at each game tour by the gradient descent Sarsa(λ) algorithm [Sutton and Barto 1998]. The output layer propagates values by an ordinary linear function. The hidden layer contains 80 neurons, propagates values by a hyperbolic tangent function, and uses a learning rate $\alpha_{hidden} = \alpha_{output} / \sqrt{n}$, where $n$ is the number of neurons of the output layer. Sarsa(λ) is tuned so as to use a learning rate $\alpha_{output}$=0.05 decaying by 10% at every evaluation, a constant exploration rate ε=0.01, and an eligibility trace λ=0.7.

### 4.1.3 Coordination and Reward Function

We tackle the issue of coordination experimentally by testing two hypotheses: (1) Can some form of limited communications between leaders of the same hierarchical level improve performance? In the RL framework, increased communications means a more complex state space. So we study this trade-off here, by including in the state description the orders (high-level tasks combined with keys locations on the map) given to same-level leaders in the previous game turn. Three leaders that are present in the higher level of the scenario used in this experiment share their previous decisions with them; (2) How best can each leader be rewarded for its order? Two simple options are considered: on one hand, a global reward function for which each leader is rewarded based on the game points scored in the current turn by his entire army (the score results from gaining control of objectives and from the losses inflicted and received); on the other hand, a local reward function where a leader is rewarded for the score points that only the units it controls obtained (taking in account objectives conquered by its group, as well as losses suffered by its group or inflicted on the opponent). The reward signal $r$ is computed at the end of each turn $t$ as the change in the game score between previous and current game turns:

$$r_{t+1} = score_{t+1} - score_t$$

where $score_{t+1}$ and $score_t$ are computed in function of conquered and lost objectives, and opponent and own army losses by:

$$score_t = \sum_{k=0}^{t} (conqObj_k - lostObj_k + oppLess_k - ownLess_k)$$

## 4.2 Learning Results

In this section, we present experimental results obtained when learning strategies for French and Russian armies in the context of the *Death in the Flèches* scenario.

### 4.2.1 French Offense

In this first experiment, our agents learn a behavioral strategy for the French army. Their goal is to maximize the score. Figure 8 shows the performance of four setups for the learning agents: with and without communication combined with global and local rewards. The learning agents have as opponent the bootstrap AI. The bootstrap AI is used both as opponent and support to leave only a small part of the decision making to be learned at any given time. The learning AI takes control over a level of the French

side, while the bootstrap AI takes control of subordinate levels, in addition to all levels of the opponent.

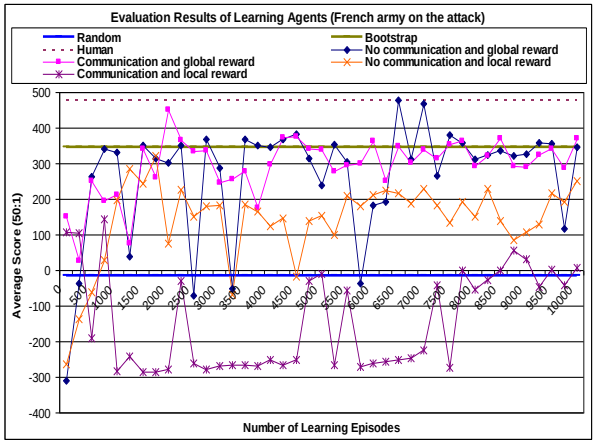


Figure 8: Performance evolution for the French army comparing learning agents with and without communication, with local and global reward, and baselines (random, bootstrap AI, human).

The performance achieved with these setups is compared with that of the bootstrap AI, of the random agent and of the human player. In the learning curves provided, each point corresponds to one evaluation phase whose value is the average score over the 50 corresponding episodes. We observe the following results:

- Several setups achieve the same performance as the bootstrap AI in only a few thousands of episodes. They correspond to the ones using a global reward (see normalized score in Figure 9). The strategy learned with these setups consists in undertaking a frontal attack and circumvent the territory dominated by the opponent through the edges of the forest in order to capture the most important objective (see Figure 10);
- Communication of orders between agents has little impact on the convergence point, but seems to help to get a smoother progression of performance;
- Local rewards do not lead to good results in these experiments. They lead to an average performance (somewhere between random and bootstrap AI) *with communication* was present, and to a very poor one *without communication*. The strategy learned by this setup shows that the agents do not cooperate in the execution of their tasks. A group of units prefers to beat a retreat instead of helping the partner in the combat against Russians since it undergoes heavy losses and never conquer scenario objectives. Consequently, the partner alone becomes hopeless against the strength and the privileged location of the Russian army.

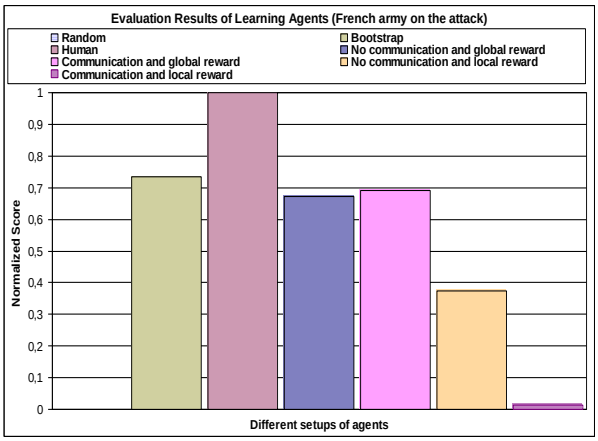


Figure 9: Normalized scores for all setups of learning agents controlling the French army and baselines agents (random, bootstrap AI, human).

These results are very interesting and significant because they show us that a global reward is the key to the attack winning when controlling the high level of the hierarchy. This makes some sense in the context of a 19th century military offensive which generally entailed at some point sacrificing numbers of soldiers in headlong charges (fresh units, a reserve, would eventually, at a later stage, overcome the defenders), a behavior which would be naturally discouraged by a local (selfish) reward.

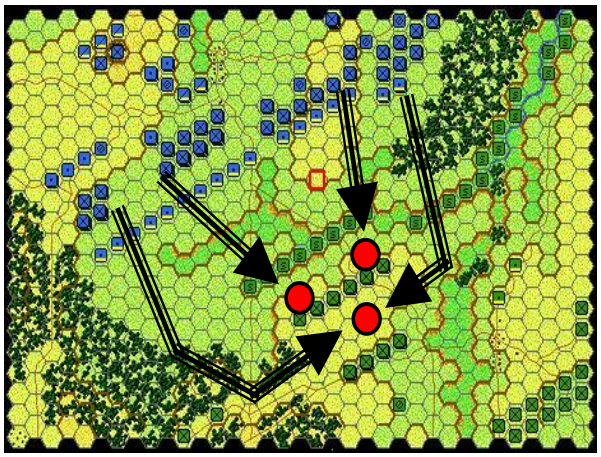

Figure 10: General outline of the strategy learned for the French army in the *Death in the Flèches* scenario. French units are blue and Russian units are green (several units can occupy the same location). Red circles are scenario objectives.

### 4.2.2 Russian Defense

In this second experiment, we reverse the roles by asking our agents to learn a strategy for the Russian army. The goal for the Russian side is *to minimize the game score*. Hence, the lowest curve is the best one. Figure 11 shows the performance of four setups for the learning agents: with and without communication combined with global and local rewards. The learning agents always have as opponent the bootstrap AI. We observe the following results:

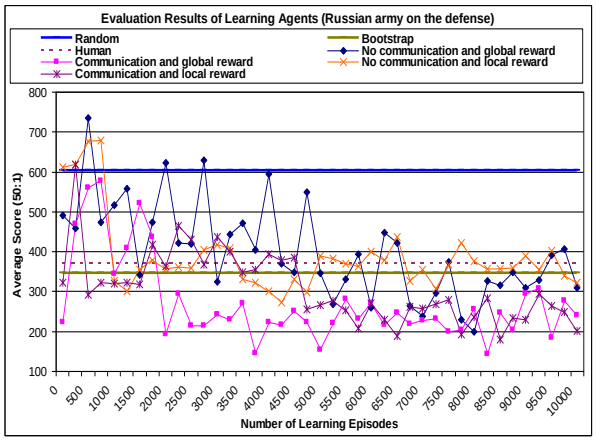


Figure 11: Performance evolution for the Russian army comparing learning agents with and without communication, with local and global reward, and baselines (random, bootstrap AI, human). Russian goal is to achieve the *lowest* game score.

- All setups obtain interesting performance (at least the same level as the bootstrap AI and the human player) with more stable curves than the attacker side (see normalized score in Figure 12). In Battleground, the defender side is always placed on a privileged location. So, defending strategies seem simpler as a result of the relatively low mobility of the simulated armies. It does not seem strange that the learning agents in the defensive obtain better performance than in the offensive.
- Communication of orders between agents increases performance significantly. It helps to get a smoother progression. Setups based on communication achieve excellent results, outperforming by far the bootstrap AI and the human player;
- A key difference here is that local rewards are not penalizing. This seems to indicate that it is not as harmful to the army if agents follow a selfish strategy while on a defensive position.

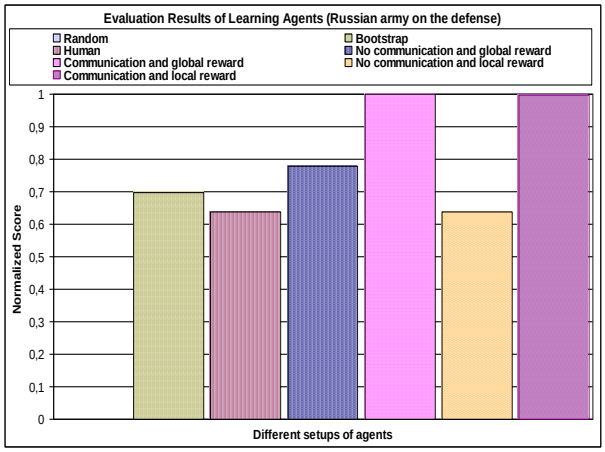


Figure 12: Normalized scores for all setups of learning agents controlling the Russian army and baselines agents (random, bootstrap AI, human).

## 5. Related Work

We can mention here some works that have applied different approaches to strategy games environments. A number of researchers have focused on single algorithms to a single aspect of the games. Guestrin et al. [2003] proposed RMDP to generalize strategic plans within the framework of multiple environments. Marthi et al. [2005] proposed Concurrent ALisp, a language that provides a natural manner to specify behavior of the multithreaded systems. Kovarsky and Buro [2006] used PDDL to explore the tactical decisions involved in building orders. Chan et al. [2007] provided planning mechanisms specialized to resource production. Balla and Fern [2009] proposed an adaptation of the UCT planning algorithm to the domain of strategy games.

Other researchers have been interested in approaches capable of playing entire games. Muñoz-Avila and Aha [2004] used HTN planning. Ponsen et al. [2006] proposed ESTG, a methodology based on evolutionary learning for automatically generating tactics with Dynamic Scripting. Sharma et al. [2007] proposed CARL, a multilayered architecture that combines Case-Based Reasoning (CBR) and RL to achieve transfer while playing against the Game AI across a variety of scenarios. McCoy and Mateas [2008] integrated multiple specialist components by incorporating expert high-level strategic knowledge. Wintermute et al. [2007] proposed SORTS, a middleware that interfaces the ORTS game engine to the Soar cognitive architecture. Navarro and Corruble [2009] employed dynamic tactical points and specific training scenarios for the learning AI for ORTS. Langley et al. [2005], Nason and Laird [2005], Hinricks and Forbus [2007], and Wilson et al. [2008] works have applied transfer learning in real-time game environments using Markov logic networks and bayesian techniques.

## 6. Conclusion and Future Work

In this paper, we described a multi-level abstraction method based on terrain analysis that adapts semi-automatically the level of detail of the state and action representations to a distributed hierarchical decision-making structure. This hierarchical structure does not require either a strategy set or an evaluation function, but rather only a set of abstract states and actions that are provided along with the ability to simulate their effects. We combined this method with RL techniques and an innovative learning scenario to construct a learning system for the automatic design of behavioral strategies that can tackle the complexity of modern strategy games.

The results obtained are very encouraging since our decision-making system achieves very good performance when controlling the higher level of the

hierarchy and compared with the scripted-based commercial AI. Only a few thousands of learning episodes were needed to achieve these results. This confirms the effectiveness of our approach, the coherence of the state and action representations designed, and validate in practice that it is possible to improve the performance of a system by learning only a part of a global strategy [Stone 2000].

We studied several setups for the learning agents by combining coordination and rewarding strategies in the framework of the some Battleground scenarios. Encouraging initial results showed the importance of using some form of communication between agents of the same hierarchical level, and global rather than local reward functions when learning attack strategies. We are now studying in more detail the issue of coordination in order to experiment more developed techniques.

More generally, we believe that terrain analysis is able to better support reinforcement learning in order to permit strategic reasoning and that our method has a relatively wide applicability and could be also used for serious games where multiple units act together on a map to achieve a common goal. Also the ideas and concepts presented here are not specific to games, and could be applied to most large multiagent simulations, especially when the agents are naturally organized in a form of hierarchical structure.

## References


BALLA, R-K, AND FERN, A. 2009. UCT for Tactical Assault Planning in Real-Time Strategy Games. *In: Proceedings of Twenty-First International Joint Conference on Artificial Intelligence*.

BLUM, A., AND LANGLEY, P. 1997. Selection of Relevant Features and Examples in Machine Learning. *Artificial Intelligence*, 97(1-2):245-271.

BURO, M. 2003. Real-Time Strategy Games: A new AI Research Challenge. *In: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*.

CHAN, H., FERN, A., RAY, S., WILSON, N., AND VENTURA, C. 2007. Online Planning for Resource Production in Real-Time Strategy Games. *In: Proceedings of the International Conference on Automated Planning and Scheduling*.

CLAUS, C., AND BOUTILIER, C. 1998. The dynamics of reinforcement learning in cooperative multiagent systems. *In: Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pp.746-752.

CORRUBLE, V., MADEIRA, C., AND RAMALHO, G. 2002. Steps Toward Building a Good AI For Complex Wargame-Type Simulation Games. *In: Proceedings of the Third International Conference on Intelligent Games and Simulation*.

DIETTERICH, T. 2000. Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *Journal of Artificial Intelligence Research*, 13:227-303.

FORBUS, K., MAHONEY, J., AND DILL, K. 2001. How qualitative spatial reasoning can improve strategy game AIs. *In: Proceedings of the AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment*.

GIUNCHIGLIA, F., AND WALSH, T. 1992. A Theory of Abstraction. *Artificial Intelligence*, 56(2-3):323-390.

GRINDLE, C., LEWIS, M., GLINTON, R., GIAMPAPA, J., OWENS, S., AND SYCARA, K. 2004. Automating Terrain Analysis: Algorithms for Intelligence Preparation of the Battlefield. *In: Proceedings of the Human Factors and Ergonomics Society*.

GUESTRIN, C., LAGOUDAKIS, M. AND PARR., R. 2002. Coordinated Reinforcement Learning. *In: Proceedings of the Nineteenth International Conference on Machine Learning*.

GUESTRIN, C., KOLLER, D., GEARHART, C., AND KANODIA, N. 2003. Generalizing Plans to New Environments in Relational MDPs. *In: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*.

HINRICHS, T., AND FORBUS K. 2007. Analogical Learning in a Turn-Based Strategy Game. *In: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*.

JONG, N., AND STONE, P. 2005. State Abstraction Discovery from Irrelevant State Variables. *In: Proceedings of Nineteenth International Joint Conference on Artificial Intelligence*.

LI, L., WALSH, T., AND LITTMAN, M. 2006. Towards a Unified Theory of State Abstraction for MDPs. *In: Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics*, pp.531-539.

MADEIRA, C., CORRUBLE, V., RAMALHO, G., AND RATITCH, B. 2004. Bootstrapping the Learning Process for the Semi-automated Design of a Challenging Game AI. *In: Proceedings of the AAAI Workshop on Challenges in Game AI*, pp.72-76.

MARTHI, B., LATHAM, D., RUSSELL, S., AND GUESTRIN, C. 2005. Concurrent Hierarchical Reinforcement Learning. *In: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*.

MCCOY, J., AND MATEAS M. 2008. An Integrated Agent for Playing Real-Time Strategy Games. *In: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*.

MUÑOZ-AVILA, H., AND AHA, D. 2004. On the Role of Explanation for Hierarchical Case-Based Planning in Real-Time Strategy Games. *In: Proceedings of ECCBR-04 Workshop on Explanations in CBR*.

NAREYEK, A. 2004. AI in Computer Games. *ACM Queue*, 1.

NASON, S., AND LAIRD, J. 2005. Soar-RL, Integrating Reinforcement Learning with Soar. *Cognitive Systems Research*, 6(1):51-59.

NAVARRO, L., AND CORRUBLE, V. 2009. Extending the Strada Framework to Design an AI for ORTS. *In: Proceedings of the 8th International Conference on Entertainment Computing*.

PONSEN, M., MUÑOZ-AVILA, H., SPRONCK, P., AND AHA, D. 2006. Automatically Generating Game Tactics via Evolutionary Learning. *AI Magazine*, 27(3):75-84.

RABIN, S. 2003. *AI Game Programming Wisdom 2*. Charles River Media.

RABIN, S. 2006. *AI Game Programming Wisdom 3*. Charles River Media.

SAITTA, L., AND ZUCKER, J.-D. 2001. A Model of Abstraction in Visual Perception. *Applied Artificial Intelligence*, 15(8):761-776.

SHARMA, M., HOLMES, M., SANTAMARIA, J., IRANI, A., ISBELL, C., AND RAM, A. 2007. Transfer Learning in Real-Time Strategy Games Using Hybrid CBR/RL. *In: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*.

STONE, P. 2000. *Layered Learning in Multi-agent Systems: A Winning Approach to Robotic Soccer*. MIT Press.

SUTTON, R. S., AND BARTO, A. G. 1998. *Reinforcement Learning, An Introduction*. MIT Press.

TESAURO, G. 2002. Programming Backgammon Using Self-teaching Neural Nets. *Artificial Intelligence*, 134:181-199.

WILSON, A., FERN, A., RAY, S., AND TADEPALLI, P. 2008. Learning and Transferring Roles in Multi-Agent Reinforcement Learning. *In: Proceedings of the AAAI-08 Workshop on Transfer Learning for Complex Tasks*.

WINTERMUTE, S., XU, J., AND LAIRD, J. 2007. SORTS: A Human-Level Approach to Real-Time Strategy AI. *In: Proceedings of the Third Artificial Intelligence and Interactive Digital Entertainment Conference*.