THESE DE DOCTORAT DE L'UNIVERSITE PARIS 6
PIERRE ET MARIE CURIE

Spécialité

INFORMATIQUE

Présentée par

CHARLES ANDRYE GALVAO MADEIRA

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE PARIS 6

Sujet de la thèse

Agents adaptatifs dans les jeux de stratégie modernes : une approche fondée sur l'apprentissage par renforcement

EXTENDED SUMMARY IN ENGLISH

Adaptive Agents for Modern Strategy Games: an Approach Based on Reinforcement Learning

Soutenue le 25 avril 2007

Devant le jury composé de :

M. Bruno BOUZY	Rapporteur	Maître de Conférences HDR à l'Université René Descartes
M. Vincent CORRUBLE	Encadrant	Maître de Conférences à l'Université Pierre et Marie Curie
M. Jean-Yves DONNART	Examinateur	Thalès Services
M. Jean-Gabriel GANASCIA	Directeur	Professeur à l'Université Pierre et Marie Curie
M. Michael LITTMAN	Rapporteur	Professeur à Rutgers University
M. Geber RAMALHO	Examinateur	Professeur à l'Universidade Federal de Pernambuco
M. Olivier SIGAUD	Examinateur	Professeur à l'Université Pierre et Marie Curie

Contents

Abstract

This thesis investigates the challenges posed by the application of reinforcement learning to modern strategy games. These games require the players to manage a high number of units placed in a very sophisticated environment so as to achieve collectively a common goal. Designing interesting solutions for such applications require tackling simultaneously several challenging issues (decision-making under uncertainty, spatial and temporal reasoning, coordination between units, etc.), each of which can represent an important research problem in itself. We propose STRADA, a novel integrated learning approach to the automatic design of ADAptive behavioral STRategies for modern strategy games. STRADA combines state-of-the-art techniques from several areas of machine learning with new ideas. In particular, it investigates two main issues: complexity reduction of the problem through decomposing the decision-making and abstracting state and action spaces. Moreover, acceleration of the process of learning from interaction by generalizing value function and bootstrapping the acquisition of experience. Solutions to these issues are combined into an efficient learning system, whose performance is demonstrated on the task of learning valuable behavioral strategies for a commercial wargame. The resulting system outperforms by far the existing commercial script-based solution.

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

1 Introduction

In this thesis, we investigate the use of reinforcement learning (RL) techniques to design efficient behavioral strategies for modern strategy games. Modern strategy games range from real-time strategy games, such as Age of Empires® (Microsoft®), to turn-based strategy games, such as John Tiller's Battleground™ series (Talonsoft®). They are characterized by huge state and action spaces, stochastic environments, and a large number of units that can act simultaneously to achieve a long-term common goal. They constitute large-scale multiagent simulations that share many characteristics with real-world problems (Schaeffer & Van den Herik, 2002; Buro, 2003) and whose complexity dwarfs that of problems typically addressed by the RL community (Madeira et al., 2004).

Consider Battleground™ (Talonsoft®), a turn-based commercial wargame that we take as case study (see Section 4). Because of its parallel nature (at each turn, *all* military units can act simultaneously), the complexity of a centralized action selection for each side 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). Regarding the configuration of a very 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 here leads to a huge state space in the order of 10^{2000} and a maneuver action space of 10^{179} concerning one army and of 10^{147} concerning the other.

This high complexity still constitutes important theoretical and practical challenges to state-of-the-art methods, techniques and algorithms from the field of RL. Typically, in a research setting, challenging issues are studied in isolation while making simplifying assumptions about other aspects of the problem. Although satisfactory solutions exist for many issues, a great challenge remains in integrating them into a single working system because properties of different techniques can be affected and altered by simultaneous applications of methods. This is the case, for example, when single-agent learning techniques are applied in a multiagent setting or when function approximation is used with RL. So, a natural question is to ask whether state-of-the-art RL techniques are up to the job of tackling the complexity of learning in modern strategy games.

Taking some inspiration from abstraction techniques, we develop some new ideas and combine them into an approach to the automatic design of adaptive behavioral strategies for modern strategy games. This approach, called STRADA, explores two main issues: (1) treatment and management of domain knowledge for decomposing the decision-making problem in a distributed fashion and abstracting state and action spaces; and (2) acceleration of the process of learning from interaction by generalizing value functions and bootstrapping the acquisition of learning experience. It is the simultaneous treatment of these issues that lets us construct a system that is able to design automatically efficient behavioral strategies for modern strategy games. The resulting system is successfully applied to Battleground™, outperforming by far the existing commercial script-based solution.

The rest of the manuscript is organized as follows. In the next section, we briefly introduce RL, discuss the challenges for its application to modern strategy games and give some directions to allow this application. Following this, we present all the aspects of the STRADA approach and briefly describe our case study, the Battleground™ wargame. We present then experiments applying our approach to Battleground™, carrying out an evaluation of various setups of our learning agents and comparing their performance to that of baseline techniques (a script-based agent and a random agent) and a human player. These setups focus especially on key issues of coordination for multi-agent learning, including inter-agent communication and the design of adequate individual reward functions. Finally, we conclude by outlining possible improvements and directions for future work.

2 Background

2.1 Reinforcement Leaning

Reinforcement learning is a general approach for learning behavioral strategies by maximizing a cumulative reward from interaction with a stochastic and unknown environment (Sutton & Barto, 1998). Markov decision processes (MDPs) present one natural way to formulate RL decision-making.

A MDP is defined as a 4-tuple ($\boldsymbol{S},\boldsymbol{A},\boldsymbol{P},\boldsymbol{R}$), in which $\boldsymbol{S} = (\boldsymbol{v_1},...,\boldsymbol{v_n})$ is a finite set of states (such as the situation of the game at a given time) defined in terms of state variables, $\boldsymbol{A} = (\boldsymbol{a_1},...,\boldsymbol{a_m})$ is a finite set of actions (what actions the player can choose from) defined in terms of actions variables, $\boldsymbol{P}$: $\boldsymbol{S}$ x $\boldsymbol{A}$ x $\boldsymbol{S} \rightarrow [0, 1]$ is a state transition probability function and $\boldsymbol{R}$: $\boldsymbol{S}$ x $\boldsymbol{A}$ x $\boldsymbol{S} \rightarrow \Re$ is a reward function defined as a real-valued bounded function. The transition probability function $\boldsymbol{P}$ and the reward function $\boldsymbol{R}$ define the dynamics of the environment where an agent makes sequential decisions. $P^{a}_{ss'}$ denotes the probability that action *a*, when executed by the agent in state *s*, transfers the environment to state *s'*. $R^{a}_{ss'}$ is the reward of taking action *a* in state *s* and observing *s'* at the next state. Transitions and rewards are generally stochastic and follow the Markov property. The goal of the RL agent is to find a policy $\boldsymbol{\pi}$: $\boldsymbol{S}$ x $\boldsymbol{A} \rightarrow [0, 1]$ which maximizes the cumulative reward over time. A policy denotes the probability that the agent takes action *a* when the environment is in state *s*.

Many RL algorithms estimate value functions, which are defined with respect to policies and reflect the expected value of the long-term accumulated reward. These algorithms determine how agents must modify their strategy to maximize the long-term accumulated reward. Sarsa (Sutton & Barto, 1998) and Q-learning (Watkins & Dayan, 1992) are examples of classical RL algorithms. They are variations of the standard dynamic programming methods that can be used on line, and do not require a model of the environment. However, these algorithms suffer from the curse of dimensionality.

Many extensions employing generalization methods were suggested in the last years in order to treat problems with large state spaces (Sutton & Barto, 1998; Santamaria, Sutton & Ram, 1998; Smart & Kaelbling, 2000; Ratitch & Precup, 2004). Generalization methods combined with reinforcement learning is referred as function approximation; it takes examples from a target function (that is, a value function) and attempts to generalize from them to construct an approximation of the entire function.

2.2 Challenges in Applying Reinforcement Learning to Modern Strategy Games

It is known that RL is very sensitive to the size of state and action spaces. So, we cannot directly repeat with modern strategy games the success obtained with TD-Gammon (Tesauro, 2002) using entire state and action spaces. In that respect, modern strategy games such as wargames are closer to real-world problems than classical games. An adequate treatment of the problem is crucial in order to propose a solution that obtains satisfactory results in a reasonable time and memory.

In this context, two central questions directly related to the complexity are mentioned (Corruble, 2000; Corruble, Madeira & Ramalho, 2002; Madeira et al., 2004). Firstly, how to represent an environment state (a situation of a game in a given time) whether one cannot do it by employing an entire situation? Secondly, how to choose coherent actions for a set of agents according to the situation? In this thesis, we study and propose some answers to these issues.

First of all, it is crucial to determine the architecture of the decision-making system. Is this as a human player who controls simultaneously a large number of units (a monolithic system controlling the behavior of several units)? Is this the individual decision-making of each unit (multi-agent distributed systems producing local and independent behaviors)? Both approaches present advantages and disadvantages.

2.2.1 Centralized approach

The main advantage of the centralized approach is that it can obtain a certain coordination of the units. It presents then a high probability to achieve optimal results. Moreover, the problem is adapted to the MDP formalism, since the environment description provides the situation of all units and the selected action is a joint action of all units (Boutilier, 1996). Hence, it is theoretically possible to directly employ RL algorithms combined with a function approximator.

In this context, a natural approach to employ function approximators would be one that generalizes state and action spaces simultaneously (Baird & Klopf, 1993; Ratitch, 2005). However, a function approximator which provides generalization to large discrete action spaces, such as those handled in Battleground™, always poses significant practical challenges to RL algorithms. So, at the current state of research, it seems impractical to use a centralized approach for Battleground™.

2.2.2 Multi-agent Distributed Approach

Another useful AI approach is that of the multi-agent distributed systems (Weiss, 2000). Each agent perceives his environment state and uses it to make a personal decision. Capacities of perception and decision-making are distributed, which avoids a combinatorial explosion.

However, the MDP formalism is not well adapted to this approach. First of all, since each agent is modeled as a local MDP which does not take into account the other agents of the environment, it does not perceive a full environment state (the environment is partially observable) (Cassandra, 1998; Kaelbling, Littman & Cassandra, 1998). Secondly, MDPs only guarantee convergence of RL algorithms to stationary environments. However, since agents learn simultaneously, they constitute uncertain elements which bring to a non-stationary environment. Consequently, the Markov property is not satisfied and the collective solutions can become sub-optimal because they depend on the local behavior of each agent and its interactions with other agents.

So, one can note that traditional methods of RL do not transpose easily to the multi-agent framework. They require an adaptation in order to allow a collective effort of all agents. This naturally poses a fundamental problem: how to coordinate individual actions of the agents (Malone & Crowston, 1994)? This need for coordination is particularly evident in multi-agents problems constituted of heterogeneous agents (each agent has a limited capacity of action) (Paquet, 2006). It is exactly the case in modern strategy games.

Solutions to problems of coordination can be divided in three principal approaches (Boutilier, 1996): conventions, communication and learning. We explore in this thesis reinforcement learning of coordination as this approach seems promising for agents evolving in complex and dynamic environments and interacting the ones with the others. Moreover, it does not require specific prior knowledge.

2.3 Reinforcement Learning of Coordination

Approaches for reinforcement learning of coordination can be classified in three main groups: (1) Multi-agent Markov decision process (MMDP); (2) emergent coordination; and (3) knowledge-based coordination.

2.3.1 Multiagent Markov Decision Process

Multi-agent Markov decision processes (MMDPs) formalize a multi-agent coordination problem. This formalism can be seen as similar to conventional MDP, except that actions are implemented in a distributed way and rewards are independent and arbitrary for each agent (Boutilier, 1996). Agents coordinate by using a probability distribution for each action of all agents. MMDPs are classified along two axes: (1) a general case MMDP (each agent *i* receives an independent reward r_i) based on Markov games (Littman, 1994, 2001; Uther & Veloso, 1997; Hu & Wellman, 1998, 2003; Tesauro, 2004); and (2) a special case of fully cooperative MMDP or *Joint-Action Learners* (*n* cooperative agents share the same reward $r_1 = \ldots = r_n$) (Claus & Boutilier, 1998; Kapetanakis & Kudenko, 2002, 2004). Although these approaches have all interesting characteristics and have contributed a lot to the domain, they pose significant practical problems since each agent needs to know the strategies adopted by all agents.

2.3.2 Emergent Coordination

In the emergent coordination approach, agents do not know the strategy adopted by other agents (*Independent Learners* in Claus & Boutilier, 1998). Agents are seen as elements of the environment, being influenced mutually only by the bias of their common environment. This approach is more general than MMDP. It allows then a larger applicability, especially to problems where a lot of agents are placed in the environment. However, coordination is more difficult here. Much work can be mentioned in that area: global and cooperative learning (Crites & Barto, 1998), local and competitive learning (Sen & Weiss, 2000), COllective INtelligence (Wolpert & Tumer, 1999; Tumer & Wolpert, 2004), learning with communication (Riedmiller & Merke, 2002). Although this approach is powerful, it also presents significant limitations when it is applied to large action spaces.

2.3.3 Knowledge-based Coordination

In the last years, the field of reinforcement learning has explored the idea that the application of prior knowledge may allow much faster learning and may indeed be essential when one needs to deal with complex real-world environments. Two main directions are followed: MDP decomposition and MDP factorization.

The principle of the resolution by decomposition is a natural idea which is often exploited in AI (Simon, 1981). In this context, many researchers have turned their attention to hierarchical methods that incorporate subtasks and state abstractions in order to combat the curse of dimensionality (Parr & Russell, 1998; Sutton, Precup & Singh, 1999; Dietterich, 2000; Andre & Russell, 2002; Makar, Mahadevan & Ghavamzadeh, 2001; Barto & Mahadevan, 2003; Marthi et al., 2005; Ghavamzadeh, Mahadevan & Makar, 2006). Hierarchical reinforcement learning deals with principled ways of exploiting temporal abstraction, where decisions are not required at each step, but rather invoke the execution of temporal-extended activities which follow their own policy until termination (Barto & Mahadevan, 2003). The crucial problem here is that this approach requires significant prior knowledge in order to decompose tasks and their value function.

Others researchers have worked on complex simulations where agent organization can be exploited to reduce the complexity of the joint action space. In this context, a promising approach that involves the use of a coordination graph for efficient distributed learning in factored MDPs was proposed (Boutilier, Dearden & Goldszmidt, 2000; Guestrin, Koller & Parr, 2001; Guestrin, Lagoudakis & Parr 2002; Guestrin et al., 2003b). The approach makes factored MDPs applicable to the control of large stochastic dynamical systems, coordinating and communicating the agents directly from its structure. In this graph, each node represents an agent, and an edge indicates that the corresponding agents have to coordinate their actions. In order to reach a jointly optimal action, a variable elimination algorithm is applied that iteratively solves the local coordination problems one by one and propagates the result through the graph using a message passing scheme. The solution is exponential with respect the width of the coordination graph instead of the number of agents. However, this approach assumes a fixed coordination graph that cannot be easily determined in modern strategy games because units coordinate with others according to their tactical situation at a given time.

3 Decomposition and Abstraction Based on Domain Knowledge: an Alternative for Modern Strategy Games

In the previous section, we stated that the use of prior knowledge can considerably help to deal with the complexity of decision-making in complex problems, accelerating remarkably the learning process. So, we take some inspiration from these approaches, especially the concept of abstraction of state and action spaces. Although there is no general method to do this, we consider that problems generally have specific structures (knowledge available) which can be exploited in order to establish efficient decomposition. For example, a simple exploitation of the rules of modern strategy games can help identifying the characteristics of their units and the organization of groups (society, civilization, army, etc.). In Age of Empires® for instance, the scenario takes place at a medieval age, with groups of units composed of archers, infantry and cavalry. In Battleground™, a scenario takes place at a Napoleonic age, with groups of units composed of infantry, artillery and cavalry. So, the decision-making can be naturally organized as a structure in which each agent has a specific role. In that case, we can define a value function for groups of agents based only on the structure of familiar elements – state and action spaces and reward function.

Moreover, we consider that modern strategy games most often offer a geographical map that can be explored. So, domain knowledge regarding military decision-making and spatial information is of crucial importance in there (Corruble, Madeira & Ramalho, 2002). It can guide us toward the purpose of abstraction, a promising approach for reducing complexity in learning problems, which consists in simplifying the level of detail of the information available (Giunchiglia & Walsh, 1992; Blum & Langley, 1997; Saitta & 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 & Stone, 2005; Li, Walsh & Littman, 2006). In this context, relevant information can be obtained by techniques known as terrain analysis.

3.1.1 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 geographic areas, together with the influences of weather, 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. 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 (Rabin 2003).

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 (Corruble, Madeira & Ramalho, 2002). Moreover, we can identify significant regions of the terrain where intelligence efforts should be focused. The military aspect of the terrain can be analyzed based on the goal of a unit, level of command, composition of the forces involved (friendly and enemies), type of operation, and equipments involved. Based on these needs, one can create various tactical decision aids by integrating terrain data together with dynamic battlefield information. Consequently, all of that information can be grouped into a dataset, where a leader can look at an integrated state representation and get only what he needs to make a decision.

3.1.2 Qualitative Spatial Reasoning

Qualitative spatial reasoning (Cohn & Hazarika, 2001) provide also techniques that can help the process of terrain analysis in modern strategy games (Forbus, Mahoney & Dill, 2001). The essence of these techniques is to make qualitative abstractions to represent continuous properties of the space with discrete systems of symbols. Using these properties, one can make predictions, diagnose and explain the behavior of physical systems in a qualitative manner without recourse to an often intractable quantitative model.

This process can be based, for instance, on partitions of the space through a combination of physical and task-specific constraints. In that case, specific locations (coordinates on the map) corresponding low-level representations and displaying one of a set of different terrain features can be ignored by high-level representations, but rather summarize them in larger objects such as zones and sub-zones of the map. Zone and sub-zone types can be represented by the dominant terrain feature in a corresponding set of locations. The main role is to determine how troops can be deployed in the quickest and most effective way.

4 STRADA: an Approach for the Automatic Design of ADAptive STRategies

In this section, we develop some new ideas and combine them with state-of-the-art techniques in order to propose an innovative approach to the automatic design of adaptive behavioral strategies for modern strategy games. This approach, called STRADA (Madeira et al., 2004; Madeira, Corruble & Ramalho, 2005, 2006), is composed of five elements: (1) decomposition of the decision-making process by using a natural hierarchical organization of groups of agents; (2) abstraction of state and action spaces by designing proper and tractable state and action representations for RL in a largely automated fashion; (3) generalization of value functions by using state-of-the-art function approximators; (4) acquisition of valuable learning experience by playing against other AI opponents; and (5) learning by steps by adopting a particular bootstrap mechanism.

4.1 Decomposing the Decision-Making Process

In modern strategy games as indeed in many real-life situations, one can see that a 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 the non-terminal nodes represent the leaders of the groups of units (see Figure 1). The ability for decision-making is distributed across specific groups, also providing a framework to guide the flow of information between them. Subordinate agents receive orders from theirs superiors and use their domain-specific knowledge and local information to make their own decisions.

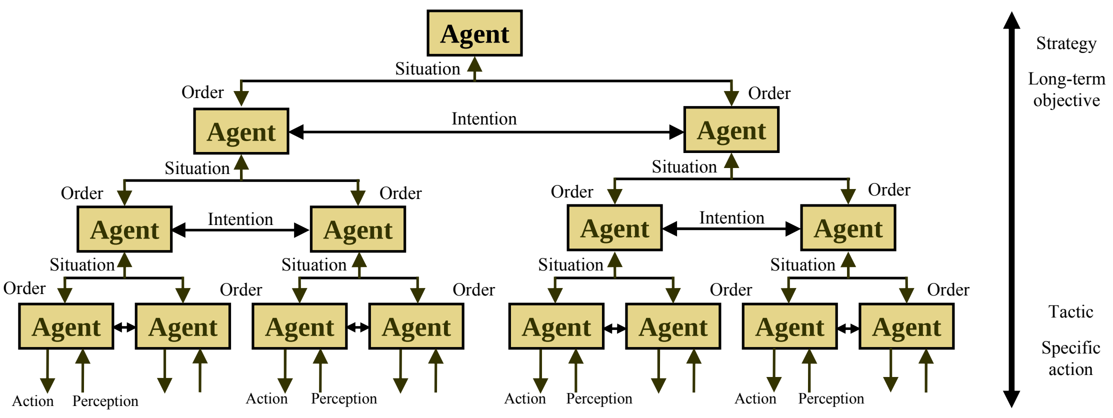

Figure 1. Hierarchical decomposition of a society, civilization, army, etc. in specialized groups of agents.

A main advantage of this structure is that it allows high-level agents (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 allows reducing the learning complexity for each group of agents (Madeira et al., 2004). Strategic decisions are made at a high level to fulfill the requirements of the groups of agents, while tactical decisions control the behavior of the small groups of units and individual low-level units. These decisions are considered as semi-independent modules and become more detailed as one goes down in the hierarchy. They can be organized in two categories: those which relate to elementary actions of the lower level (fire, turn, move, etc.) and those which combine elementary actions to allow high-level behaviors for groups (protected movement, concentrated attack, monitoring, etc.).

4.1.1 Communication established between agents in the hierarchy

However, the number of low-level units to be controlled and the intensity of interaction between their strategies increase dramatically when learning strategies for lower levels of the hierarchy. This requires some degree of coordination between the leaders in order to provide a mechanism in which the units work as a team (Tambe & Zhang, 2000). This issue is essential and subject to theoretical and empirical studies in the fields of multi-agent learning (Claus & Boutilier, 1998; Guestrin, Lagoudakis & Parr, 2002).

In this context, hierarchies can provide coordination mechanisms by modeling communication between agents as follows: (1) between a high-level agent and its subordinates; and (2) 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.

4.1.2 Structure of value function

The value function estimated by the RL algorithm can be directly represented in the hierarchy. Each node contains a local value function which represents the estimated values of the actions projected in the action space of the units belonging to this node. Local value functions of child nodes are combined towards parent nodes. So, the global computation of estimated values of actions could be carried out by engaging low-level actions for all nodes, then by propagating values from the lower level to the highest level of the hierarchy. The value obtained with the root would represent the value of the collective action of all units. However, this “bottom-up” approach would require us to learn the fully hierarchy simultaneously, certainly demanding a significant time of learning.

The STRADA approach uses another mechanism, detailed in the next sections, in which the hierarchical decomposition of specialized groups of agents has five main purposes: (1) it organizes the decision-making process; (2) it limits and guides communication between groups of agents (agents interact vertically, and horizontally inside their group); (3) it provides a guide for the abstraction of state and action spaces; (4) it provides a structure for the definition of value function; and (5) it allows learning by stages.

4.2 Abstracting State and Action Spaces

It seems essential that high-level agents be placed in more abstract state and action spaces in order to make strategic decisions. They do not need to be overwhelmed by the intractably large number of possibilities which a centralized agent would have to consider if it possessed all the information pertaining to individual low-level units. They must receive only strategic information from main groups of units, make strategic decisions founded on this information and give orders to their subordinates. Subordinates perceive information at another level of detail (less strategic and more tactic) and act in a relatively independent way, while being in harmony with the orders provided by their superiors.

Taking inspiration from concepts of terrain analysis and qualitative spatial reasoning techniques, we designed an algorithm for the semi-automatic generation of adequate state and action representations for agents of our hierarchical structure of groups. This algorithm analyzes the terrain by using detailed game scenario information, abstracts it in order to generate high-level topological concepts and generates representations which will be used as bases for the decision-making. This abstraction process is carried out in two main steps: abstraction of the action space and abstraction of the state space.

4.2.1 Abstraction of the Action Space

Tactical action spaces of modern strategy games are generally 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 propose to abstract both variables.

On one hand, we could gather a certain number of elementary maneuver tasks and combine steering parameters such as speed, discretion, safety and mass movement concentration to define high-level maneuvers (monitoring, exploration, occupation, etc.) (Corruble, Madeira & Ramalho, 2002). Nevertheless, high-level maneuvers are not yet defined automatically by our algorithm.

On the other hand, we explore the map in order to find key locations. A *key location* is any position on the map whose control is likely to give distinct military advantage to the force that holds it. For instance, a river is an obstacle that impedes or prevents the maneuver of forces. Finding key locations such as bridges is fundamental since it lets the leader make inferences about possible degree of vulnerability of friendly forces to enemy attacks. Our algorithm uses key map locations in order to reduce considerably the size of the action space. Figure 2 illustrates the identification of key locations on a real map to demonstrate the generality of our approach. As a result, once high-level maneuvers are given and key locations are identified, they are combined to generate the strategic action space.

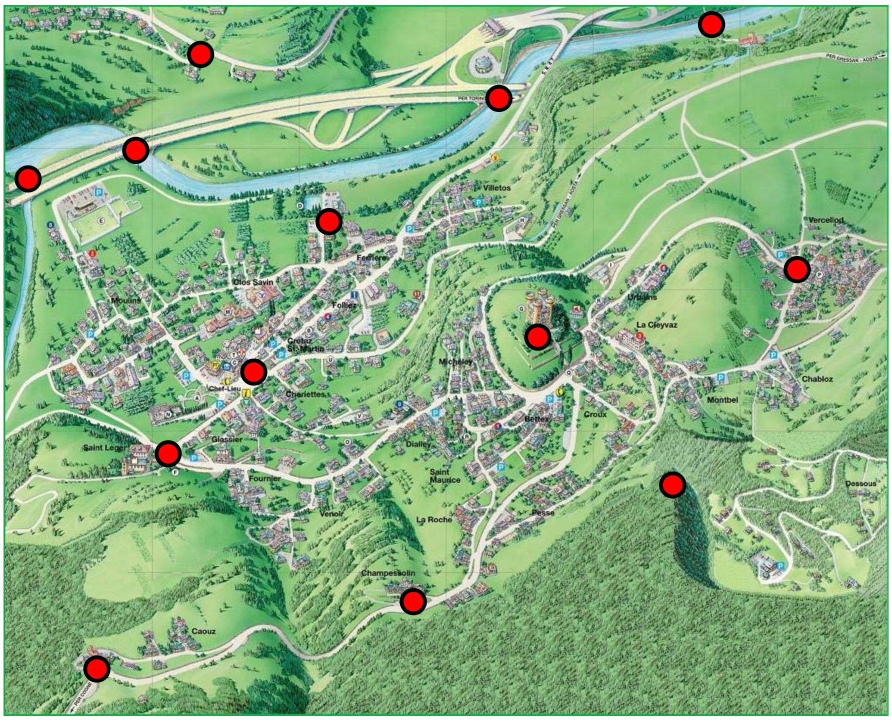

Figure 2. Identification of key locations on a real map. Bridges on the river, the castle on the top of the hill, the top of the mountain, large crossroads in the center town, crossroads of access for the city, are all good examples of key locations.

4.2.2 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, and levels of strength, fatigue, quality, mobility and ammunition of the group. Actually, variables used to compose summaries are not yet identified automatically by our algorithm.

On the other hand, the key locations identified previously in the abstraction of action spaces are used as seeds to be expanded progressively by adding neighboring cases until a full partitioning of the map into strategic zones has been reached. A *strategic zone* is a region of the map where terrain information is gathered in order to provide large-granularity information.

It is associated to a category of mobility (easy mobility, restricted mobility or no mobility) according to the nature of the terrain. Figure 3 illustrates strategic zones generated using as seeds the key locations identified on the real map of Figure 2. Such information, combined with a summary state of friendly and enemy units placed on each zone, can provide more intelligent movement on the terrain. Although type and characteristics of strategic zones can change from a game to another, the idea here is always the same because we take into account only a global vision of the terrain.

The state representation is always composed of two main groups of variables where leaders can get only the information they need to make relevant decisions: 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.

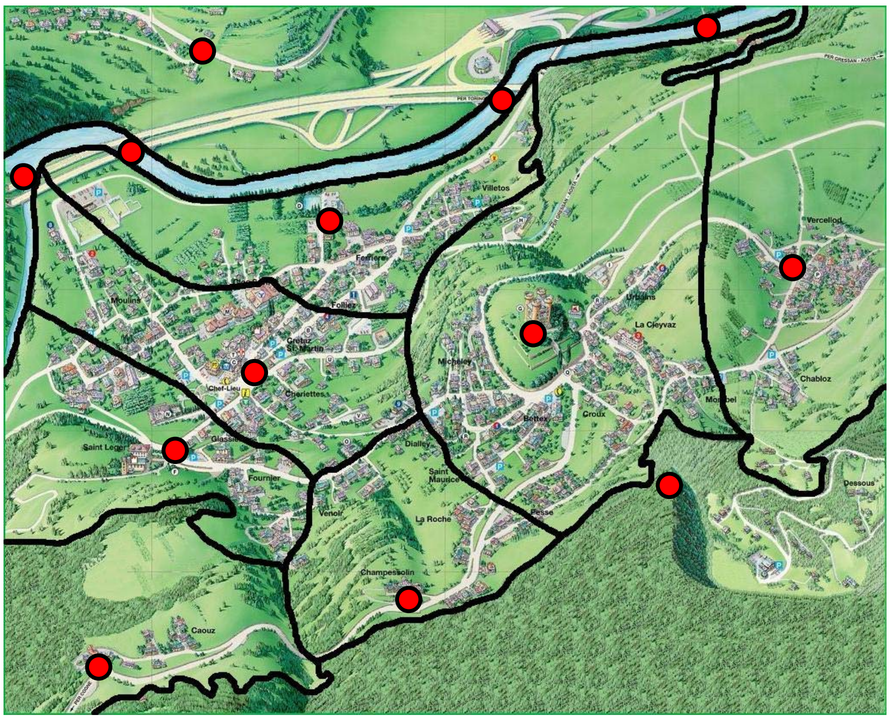

Figure 3. Identification of key locations and strategic zones on a real map. The strategic zones are generated by propagating key locations in order to aggregate adjacent locations, taking into account physical constraints of the terrain.

4.2.3 Algorithm Definition

Our algorithm, named ***generateStateActionRepresentation*** (see Figure 4) takes into account some features of a given scenario: objectives, fixed objects (ease of access and obstacles), and a two-dimensional map topography (the map locations are organized as a grid – squares, hexagons, etc.). In addition to this, it takes also as input a set of players, a set of high-level maneuvers, a set of variables describing the situation of a group, and a set of variables describing the situation of units placed on zones. The algorithm is composed of four steps as described below.

```
generateStateActionRepresentation( Objectives, Objects, MapLocations, Players,
                                   HighLevelManeuvers, GroupDescription, ZoneDescription )
{
  KeyLocations $\leftarrow$ identifyKeyLocations( Objectives, Objects, MapLocations )
  StrategicZones $\leftarrow$ identifyStrategicZones( KeyLocations, MapLocations )
  StrategicZones $\leftarrow$ fusionZones( StrategicZones, MapLocations )
  ActionRepresentation $\leftarrow$ HighLevelManeuvers x KeyLocations
  ZoneRepresentation $\leftarrow$ ZoneDescription x StrategicZones x Players
  StateRepresentation $\leftarrow$ GroupDescription $\cup$ ZoneRepresentation
  return ActionRepresentation and StateRepresentation
}
```

Figure 4. Sketch of algorithm for terrain analysis and abstraction of state and action spaces.

4.2.3.1 Identification of Key Locations (see Figure 5)

Locations, fixed objects and objectives on the map are evaluated by the ***isRelevantLocation*** function to identify those which are likely to give advantage to units placed on them (key locations). Then, adjacent key locations identified are aggregated in zones by the ***fusionZones*** function (see Figure 7). Finally, a location in each zone (that is considered the most relevant by the ***identifyTheMostRelevantLocation*** function) goes into to the list of key locations that is used to generate the abstracted action space later on.

```
identifyKeyLocations( Objectives, Objects, MapLocations )
{
  TacticalZones $\leftarrow \varnothing$
  for all $o_i \in$ ( Objectives $\cup$ Objects ) do
    if isRelevantLocation( $o_i$ ) $\wedge$ ( $\{o_i\} \not\subset$ ZonesTactiques ) then
      Zone $\leftarrow \{o_i\}$
      TacticalZones $\leftarrow$ TacticalZones $\cup$ {Zone}
    end if
  end for
  for all $c_i \in$ MapLocations do
    if isRelevantLocation( $c_i$ ) $\wedge$ ( $\{c_i\} \not\subset$ TacticalZones ) then
      Zone $\leftarrow \{c_i\}$
      TacticalZones $\leftarrow$ TacticalZones $\cup$ {Zone}
    end if
  end for
  TacticalZones $\leftarrow$ fusionZones( TacticalZones, MapLocations )
  KeyLocations $\leftarrow \varnothing$
  for all $Z_i \in$ TacticalZones faire
    case $\leftarrow$ identifyTheMostRelevantLocation( $Z_i$ )
    KeyLocations $\leftarrow$ KeyLocations $\cup$ {case}
  end for
  return KeyLocations
}
```

Figure 5. Sketch of algorithm for identifying key locations.

4.2.3.2 Identification of Strategic Zones (see Figure 6)

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 (state verified by the ***isMarkedLocation*** function) of this terrain type is used as seeds to create a new strategic zone and to be then expanded progressively by adding adjacent locations of the same terrain type using the ***propagateZoneSameTerrain*** recursive function. Following this, key locations identified in the prior step are also used as seeds to be expanded progressively by adding non-aggregated adjacent locations using the ***propagateZones*** function until a full partitioning of the map into strategic zones has been reached. The expansion rate of each location here 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 case to a zone. It is computed by several criteria: the movement cost of the terrain type, the eventual obstacles and eases of access, as well as the variation of altitude. The accumulated weight of aggregation is a parameter that indicates the probability to add a location to a zone. It is initialized to zero and it is reinforced after each aggregation failure. Finally, if needed, "lost" strategic zones are created if there still are non-aggregated locations 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.

```
identifyStrategicZones( KeyLocations, MapLocations )
{
  RestrictedZones ← $\emptyset$
  for all $c_i \in$ MapLocations do
    if $\neg$ isMarkedLocation( $c_i$ ) $\wedge$ isTerrainOfRestrictedMobility( $c_i$ ) then
      markLocation( $c_i$ )
      Zone ← $\{c_i\}$
      Zone ← propagateZoneSameTerrain( Zone, MapLocations )
      RestrictedZones ← RestrictedZones $\cup$ {Zone}
    end if
  end for
  StrategicZones ← $\emptyset$
  for all $p_i \in$ KeyLocations do
    if $\neg$ isMarkedLocation( $p_i$ ) then
      markLocation( $p_i$ )
      StrategicZones ← StrategicZones $\cup$ $\{\{p_i\}\}$
    end if
  end for
  StrategicZones ← propagateZones( StrategicZones, MapLocations )
  LostZones ← $\emptyset$
  for all $c_i \in$ MapLocations do
    if $\neg$ isMarkedLocation( $c_i$ ) then
      markLocation( $c_i$ )
      Zone ← $\{c_i\}$
      Zone ← propagateZoneSameTerrain( Zone, MapLocations )
      LostZones ← LostZones $\cup$ {Zone}
    end if
  end for
  StrategicZones ← StrategicZones $\cup$ RestrictedZones $\cup$ LostZones
  return StrategicZones
}
```

Figure 6. Sketch of algorithm for identifying strategic zones.

4.2.3.3 Fusion of Strategic Zones (see Figure 7)

Strategic zones are evaluated by the ***isRelevantZone*** function for detecting those in which size is relevant when compared to the map size. Small zones are absorbed by adjacent zones of larger boundary.

```
fusionZones( Zones, MapLocations )
{
  for all $Z_i \in Zones$ do
    if $\neg$ isRelevantZone( $Z_i$, MapLocations ) do
      AdjacentZones $\leftarrow \emptyset$
      for all ( $Z_j \in Zones$ ) $\wedge$ ( $i \neq j$ ) faire
        if isAdjacentZone( $Z_i$, $Z_j$ ) then
          AdjacentZones $\leftarrow$ AdjacentZones $\cup \{Z_j\}$
        end if
      end for
      ZoneOfLargerBoundary $\leftarrow$ getZoneOfLargerBoundary( AdjacentZones, $Z_i$ )
      Zones $\leftarrow$ Zones $\setminus$ ( {ZoneOfLargerBoundary} $\cup \{Z_i\}$ )
      MergedZone $\leftarrow$ ZoneOfLargerBoundary $\cup Z_i$
      Zones $\leftarrow$ Zones $\cup$ {MergedZone}
    end if
  end for
  return Zones
}
```

Figure 7. Sketch of algorithm for fusion of zones.

4.2.3.4 Generation of State and Action Representations (see Figure 4)

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

4.3 Generalizing value functions

It is well known that RL systems must use parameterized function approximation in order to generalize between similar situations (Sutton & Barto, 1998). This is crucial to obtain satisfactory results in a reasonable time and memory for MDPs with large or continuous state spaces. However, function approximation in the context of RL is harder than in the classical supervised learning setting (Thrun & Schwartz, 1993; Ratitch & Precup, 2002) and still present important difficulties when applied to complex problems. In fact, in RL, the target function (that is, the value function) is estimated gradually and thus appears to be non-stationary, whereas in supervised learning, many techniques assume a stationary target function. Moreover, the stochastic characteristics of the environment of modern strategy games and the exploration process may introduce variability into the training data.

Although the properties of RL algorithms using function approximation are still not fully understood, and despite the absence of general theoretical convergence guarantee, in practice, the use of function approximators has already proven quite successful in handling some realistic domains (Crites & Barto, 1996; Santamaria, Sutton & Ram, 1998; Tesauro, 2002; Coulom, 2002). So, the STRADA approach employs function approximators in order to complete its basic structure. In this context, linear and non-linear function approximators (Sutton & Barto, 1998) have been considered here.

4.4 Generating Valuable Learning Experience

Once we have developed the foundation of our decision-making system, we tackle the issue of acquisition of experience in the process of learning from interaction. In general, experience can be gathered in all the situations which constitute learning opportunities. Games are a dream application domain for learning techniques as generating experience could not be easier: one just has to play. However, we look for a mechanism which generates worthy experience to speed up the learning process.

The first option would be to have our system play against a good human player. Unfortunately, it is not viable for complex games as it would require the human player to play thousands of games (each game taking hours if not days) against an AI that would initially be a terrible player. Another data acquisition method is self-play (Kaelbling, Littman & Moore, 1996). However, the amount of time needed to learn a good strategy could be impractical here. Regarding the case of TD-Gammon (Tesauro, 2002), it took 6 millions trials to learn a great strategy by self-play. For our case study, the Battleground™ wargame, it would take orders of magnitude more because of the increased complexity. Yet 6 millions of games by self-play would already take over 30 years to simulate in a Pentium® 4 / 2.8GHZ / 1GB if we consider a simple scenario.

Therefore, we opt for another path: it consists in using “hand-made” agents to play and learn against. In this context, a basic agent composed principally of tactical (or low-level) actions can be built relatively easily, designed using techniques such as rule bases. Then, we let our system (called the *learning AI*) play and learn against this “hand-made” AI (called *bootstrap AI*) (Madeira et al. 2004). However, experience acquired by playing always against the same opponent can produce a specific strategy which does not work very well when playing against other opponents. So, after learning against the bootstrap AI, we can consider other options such as self-play in order to improve the learnt strategy.

4.5 Accelerating the Learning Process

Once defined an adequate opponent model to play against, we configure the learning scenario with a particular mechanism. We consider that even with this adequate opponent, the learning AI could not address simultaneously the entire hierarchy decision-making of its side. In fact, learning simultaneously a strategy for all levels of the hierarchy is simply not feasible as a result of the extremely high dependence of the global performance on the performances of individual levels of the hierarchy. Indeed, learning to control an entire camp at once remains a very complex task to tackle (Stone, 2000), since failures are difficult to interpret as they can result from mistakes from any level of the hierarchy (Madeira et al., 2004).

For this reason, we use another type of bootstrap mechanism by again taking advantage of the hierarchical decomposition (Madeira et al., 2004). This mechanism articulates some form of incremental learning inspired from various results (Stone, 2000; Dutech, Buffet & Charpillet, 2001; Whiteson & Stone, 2003). It lets the learning AI take only partial control over its side (one or more levels), while the bootstrap AI takes control of the subordinate levels of the learning AI side, in addition to the full control of the opponent side. This allows using the bootstrap AI as support in order to leave only a small part of the decision making process to be learned at any given time. For instance, we could apply this mechanism by learning strategies for the highest level of the hierarchy first and then progressively going down, that is, the learning AI chooses high-level orders for the subordinates it is in charge of, after which the bootstrap AI is used to implement these orders at the lower levels (see Figure 8).

This is an important point in our approach because it allows incremental learning (top-down approach) of strategies with increasingly refined levels of specificity and detail. Then, one concentrates on the learning of the higher level (strategic decisions), whose tactical low-level decisions are made. Although more than one cycle of learning be probably needed before reaching a satisfying performance level, low-level units become increasingly efficient on their tasks and the global learning effort is improved.

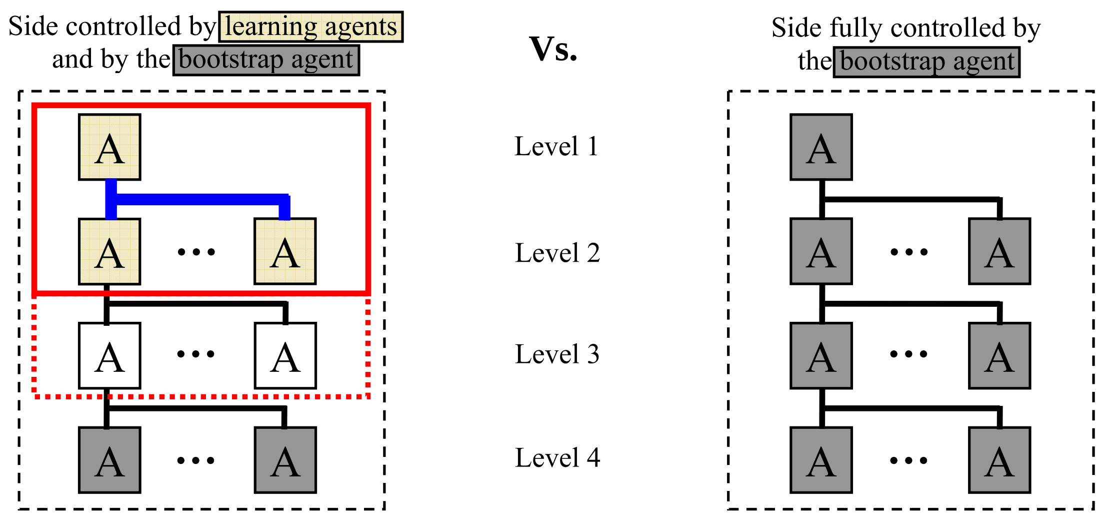
Side controlled by learning agents and by the bootstrap agent
Vs.
Side fully controlled by the bootstrap agent
Level 1
Level 2
Level 3
Level 4
A
A
A
A

Figure 8. Scheme of the STRADA approach for the decision-making for each side. First of all, learning agents learn to select orders for the level 1 of the hierarchy. Then, based on the strategy learned for level 1, they learn to select orders for the level 2. This process goes on until the level where the selection of orders is learned by agents which their subordinates are controlled by the bootstrap agent.

5 Case study

In this section, we introduce our case study, the Battleground™ wargame, and apply the elements of the STRADA approach to it.

5.1 Battleground™: a "Real-World" Problem from the Game Industry

5.1.1 Description

Battleground™ (John Tiller Games / Talonsoft®) is a turn-based commercial game that simulates the confrontation between two armies (zero-sum two players) at historical battlefields with detailed maps (see http://home.hiwaay.net/~tiller/). Its scenarios model units which move on maps composed of hundreds or thousands of hexagons (see Figure 9). These units are defined by numerous characteristics (type, strength, fatigue, quality, formation, facing, morale, mobility, ammunition and visibility), some of which are dynamic. They are organized in a military hierarchy (army, corps, division, brigade and front-line) and can represent troops (battalions of infantry, regiments of cavalry, and batteries of artillery), skirmishers, individual leaders and supply wagons.

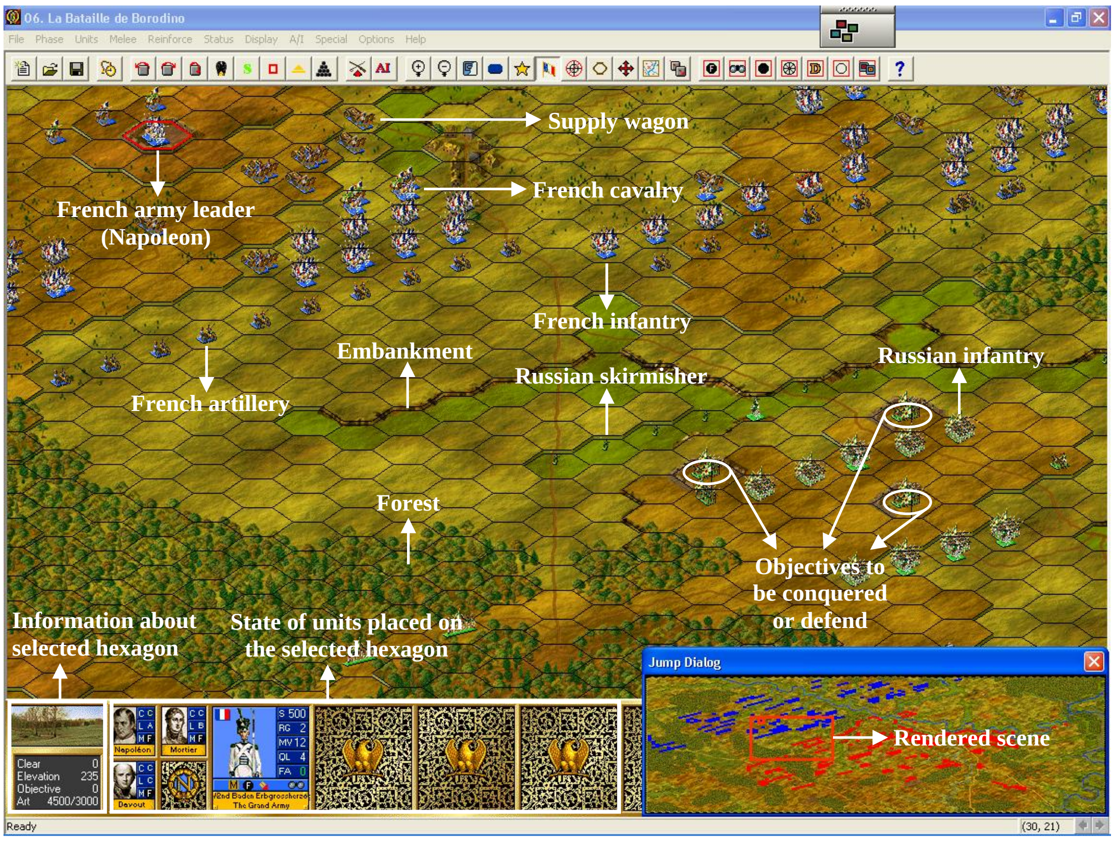

Figure 9. A screenshot of the Battleground™ 6 "Napoleon in Russia" game. It simulates the confrontation between the French (blue units) and Russian (green units) armies in the context of the battle of Borodino in 1812.

Each hexagon on the map contains a number of features such as terrain type (clear, forest, marsh, rough, water, etc.), movement cost for each type of unit, effect on the unit combat, elevation, eases of access (bridge, road, path, etc.) and obstacles (village, river, fort, embankment, etc.). Each of the unit's characteristics and hexagon's features has an associated effect on the unit's abilities to move or fight.

The combat resolution is a stochastic process whose result depends on a strength ratio between attacker and defender, as well as many other parameters for each side. The goal in each scenario is to either capture or hold specific objectives (hexagons), while trying to eliminate as many of the opponent's units as possible as well as preserving its own. A game is won by the army that scores the highest number of points in a given amount of turns.

Battleground™ provides an option to play against the computer. This game AI was developed using script-based techniques. It works relatively well in the case of simple Battleground™ scenarios where strategic decision-making is not so crucial. However, in the case of more complex game scenarios, its loopholes can be easily exploited by a good human player. The option of using ad-hoc techniques with carefully hand-crafted military knowledge has usually been the most common course of action in the game industry, though it often leads to high development costs and low-quality automated opponents (Rabin, 2003; Nareyek, 2004).

5.1.2 Modeling Battleground™ in a MDP Framework

Battleground™ maps fairly directly onto the discrete-time, episodic, reinforcement-learning framework. We can model it, in a first stage, with a centralized perspective through an MDP (***S***,***A***,***P***,***R***). Leaders give orders in a sequential manner at the beginning of each turn, and front-line units perform these orders in a number of phases. When a turn ends, another begins, giving rise to a series of turns.

The state space ***S*** is characterized by multiple variables describing the situation of a game scenario. It includes type, quality, formation, facing, location on the map, and levels of strength, fatigue, mobility, visibility, ammunition, disorder and morale, for each army unit (friendly and enemy).

The action space ***A*** is composed of a set of elementary actions (or tactical orders). Elementary actions concern combat and short-term maneuver orders that are performed by the lower level (front-line) units of the military hierarchy. They correspond to single movement, charge of cavalry, firing, and body fighting, associated with a map location.

The state transition probability function ***P*** is partly determined by the rules of the game (the rest depending on the opponent's strategy), though RL can be applied even without explicitly knowing it because RL algorithms are designed to learn directly from experience.

The reward function ***R*** can be designed based on the score of the game, itself computed by taking into account the values of each scenario objective and the losses to each side.

Consider the case of the simple Battleground™ scenario discussed previously in the introduction, called *Death in the Flèches* (see Figure 10). It simulates a stage of the battle of Borodino (1812) between the French army of Napoleon and the Russian army of Kutuzov. The French army contains 101 front-line units, the Russian army contains 83 and 3 objectives are placed on a small map of 700 (35x20) hexagons. The combinatory explosion here leads to a huge state space in the order of 10^{2000}. Moreover, each unit may move to 60 different locations on average at each turn (depending on the unit type and terrain) and may combat any of 20 enemy units on average (depending on the unit type, weapons and visibility conditions). Considering this decision-making problem with a centralized perspective (a unique MDP), the complexity grows exponentially with the number of units (if there are U units, each with A possible actions, the resulting branching factor is A^U). Then, this would lead to a maneuver action space of $60^{101} = 10^{179}$ and a combat action space of $20^{101} = 10^{131}$ to the French army, and a maneuver action space of $60^{83} = 10^{147}$ and a combat action space of $20^{83} = 10^{107}$ to the Russian army. As a consequence of the complexity encountered here, designing strategies and determining what effects decisions have towards the goal of the game is a very hard problem.

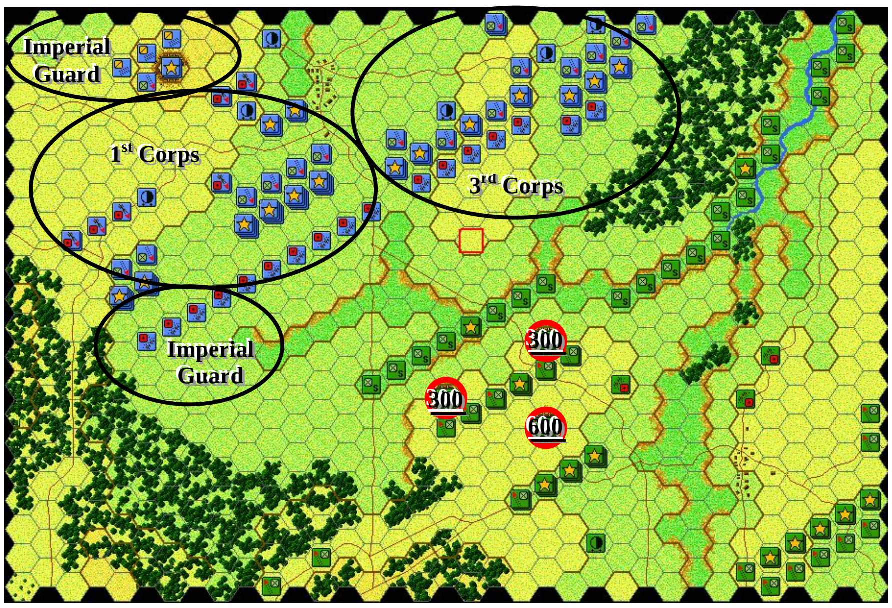

Figure 10. A 2D screenshot that shows the starting configuration of the Death in the Flèches scenario. This scenario is composed of 101 French front-line units 83 Russian front-line units placed on a map arranged in 700 (35x20) hexagons (a hexagon can be occupied by several units). The mission of the French army is to conquer 3 objectives on the map (those marked by circles with respective scores points) which are dominated by the Russian army at the beginning of the game. A complete game is played on 10 turns. The 1st corps is conducted by the Marshal Davout, the 3rd corps is conducted by the Marshal Ney and the imperial guard is conducted by the Marshal Mortier.

5.2 Applying STRADA to Battleground™

Now that Battleground™ has been introduced, we describe the application of all elements of STRADA to it. In order to ascertain the generality of our approach with more confidence, we test it on two game scenarios: (1) the *Death in the Flèches* scenario introduced previously; and (2) a more complex scenario, called *The Shevardino Redoubt*, where strategic decision-making must be more carefully designed.

5.2.1 Hierarchical Structure of Decision-Making

Given that Battleground™ units constitute armies, we take inspiration from the historical model of the military hierarchical structure of command and control. This model proposes a natural organization of agents where the decision-making is carried out on several levels from strategic to tactical decisions.

The hierarchy reproduced by Battleground™ is composed of five levels (army, corps, division, brigade and front-line) as described in Figure 11. Front-line units are placed physically on the ground and are asked to carry out orders provided by their superiors. According to the STRADA approach, this hierarchical structure guides us to abstract state and action spaces to a level of detail adapted to each agent. Moreover, it is designed in a fully distributed fashion, that is, instead of learning a unique strategy for a leader who knows all orders given for its subordinates, our system learns several strategies at the same hierarchical level, each one for an agent who gives orders for a specific subordinate. Consequently, different techniques to coordinate agents controlling the same hierarchical level can be tested.

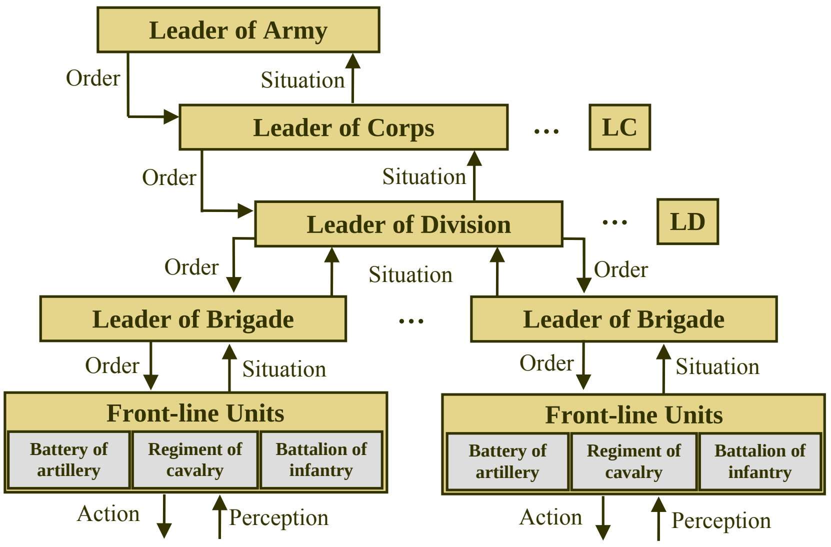

Figure 11. Military hierarchical structure of command and control of Battleground™.

5.2.2 Abstraction of State and Action Spaces

In this section, we apply our abstraction algorithm to Battleground™.

5.2.2.1 Death in the Flèches Scenario

To design a strategic action space for the *Death in the Flèches* scenario, we choose five high-level maneuvers 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 in each zone (see Figure 12).

Our abstraction algorithm analyzes the map topography to determine key locations (villages, forts, bridges, top of hills, scenario objectives, etc.) so as to abstract the action space. Then, it seeks hexagons in which associated terrain type is of restricted mobility (forests, rivers, etc.) to generate the first strategic zones on the map. It then propagates the identified key locations based on several criteria (movement cost of the terrain, ease of access, obstacles and elevation) in order to aggregate adjacent hexagons. This process generates other strategic zones which can be merged.

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 12. Variables selected to describe the state of groups and zones for the *Death in the Flèches* scenario.

The execution of our algorithm obtains as result 8 key locations and 6 strategic zones on the map (see Figure 13). 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 players).

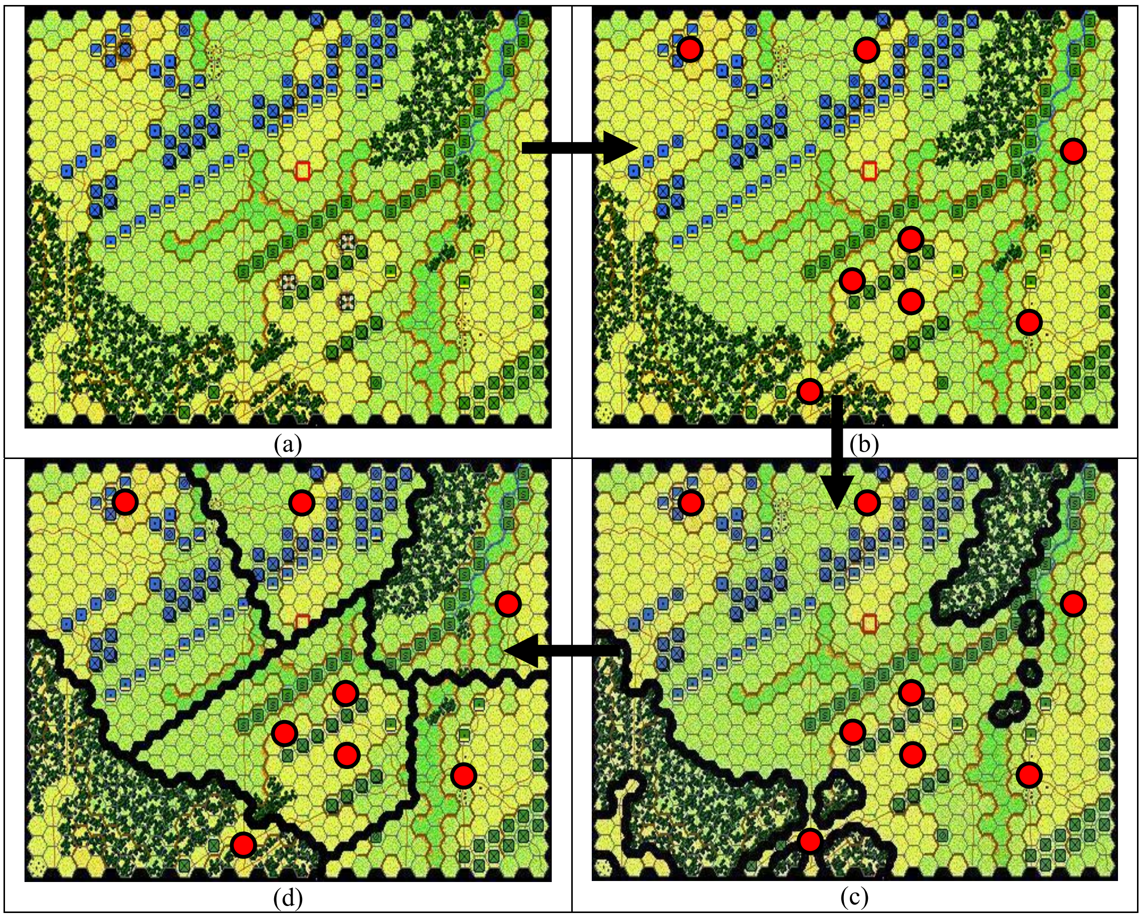

Figure 13. Execution of the algorithm of terrain analysis for the *Death in the Flèches* scenario: (a) map before terrain analysis; (b) map after identification of key locations; (c) map after determination of restricted zones; (d) map after determination of the strategic zones by a progressive expansion of key locations and fusion of adjacent zones.

In addition, high-level maneuvers are associated with key locations on the map to compose an abstracted action space of 33 strategic actions (4 high-level maneuvers x 8 key locations + *Wait order*).

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 previously (state and action spaces in the order of 10^{2000} and 10^{179} respectively).

5.2.2.2 The Shevardino Redoubt Scenario

To design a strategic action space for *The Shevardino Redoubt* scenario (see Figure 14), we choose the same five high-level maneuvers used for the *Death in the Flèches* scenario. However, to design a strategic state space, we choose 20 variables to describe the situation of groups and 2 variables to describe the situation of units in each zone (see Figure 15). This representation takes into consideration the increased complexity of this scenario. The map is larger and a game is longer than in *Death in the Flèches*, scenario objectives are isolated, half of a game is played in darkness (mobility and visibility are restricted, which makes maneuver and combat much more difficult).

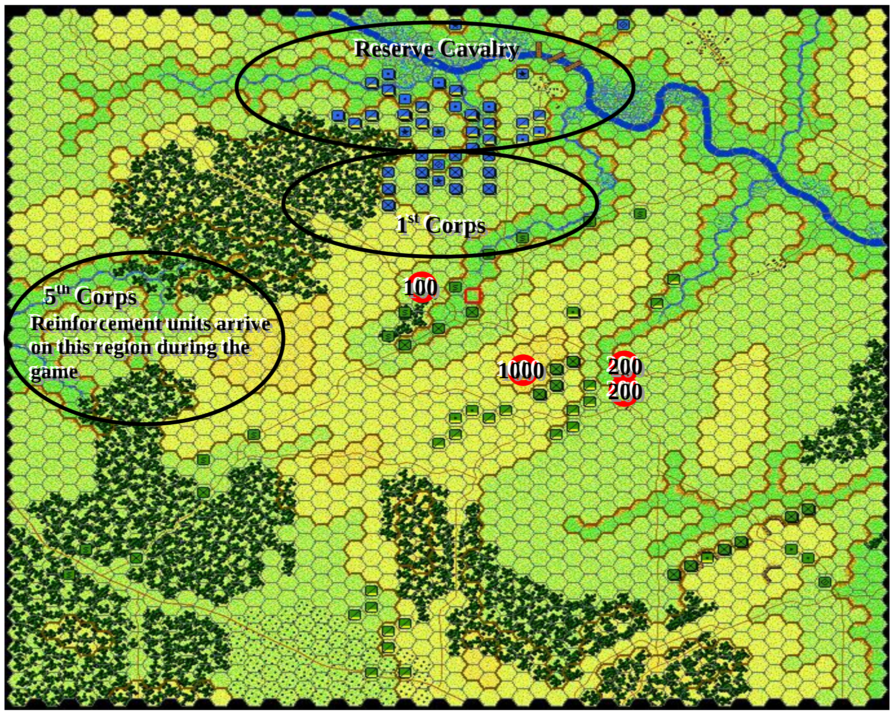

Figure 14. A 2D screenshot that shows the starting configuration of *The Shevardino Redoubt* scenario. This scenario is composed of 117 French front-line units 64 Russian front-line units placed on a map arranged in 2184 (52x42) hexagons (a hexagon can be occupied by several units). The mission of the French army is to conquer 4 objectives on the map (those marked by circles with respective scores points) which are dominated by the Russian army at the beginning of the game. A complete game is played on 18 turns. The 1st corps is conducted by the Marshal Davout, the 5th corps is conducted by the Marshal Poniatowski and the reserve cavalry is conducted by the Marshal Murat.

Group description (20 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);
- Morale, disordered, and ammunition levels (3 variables);
- Command and leadership rating levels of leaders (2 variables);
- Friendly and enemy visibility (6 variables);
- Game turn (1 variable).

Strategic zone description (2 variables)
- Unit strength and fatigue levels (2 variables).

Figure 15. Variables selected to describe the state of groups and zones for *The Shevardino Redoubt* scenario.

The execution of our algorithm of terrain analysis obtains as result 12 key locations and 12 strategic zones (a river included) on the map (see Figure 16).

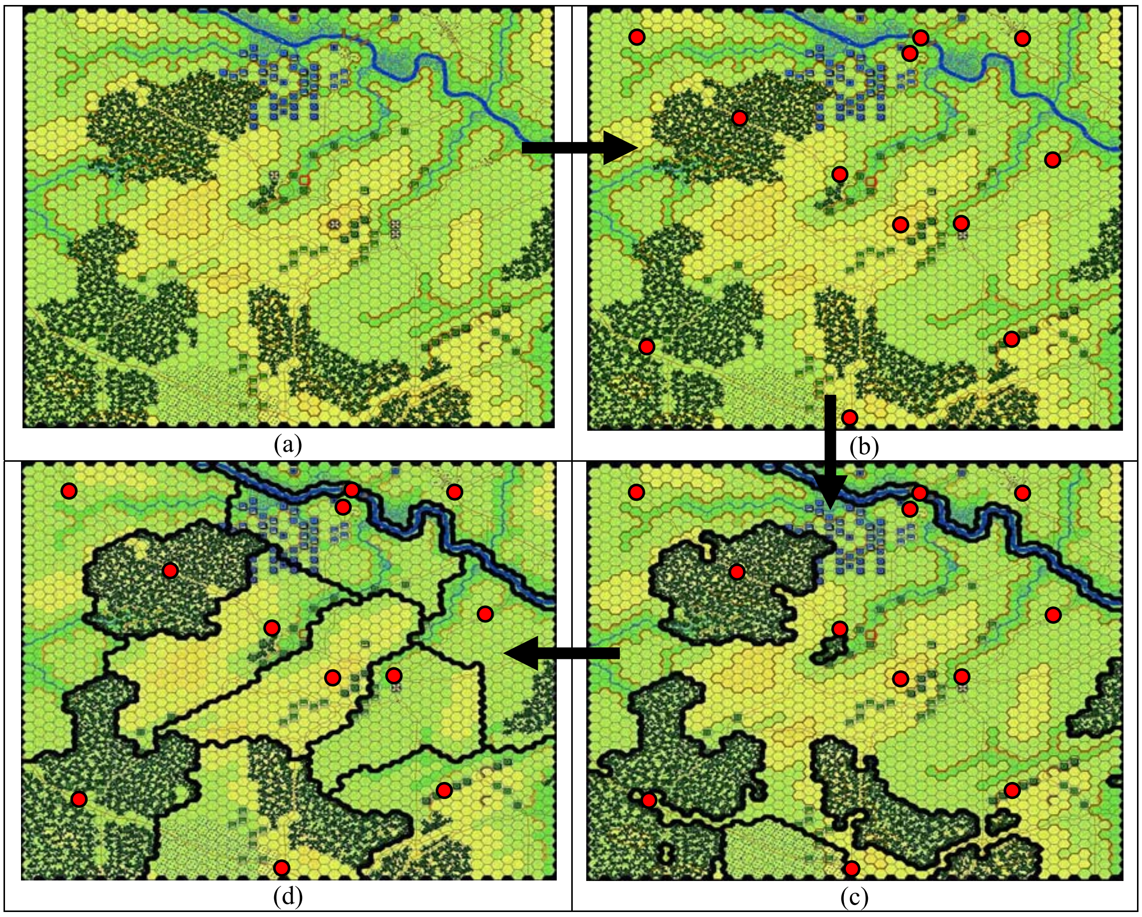

Figure 16. Execution of the algorithm of terrain analysis for *The Shevardino Redoubt* scenario: (a) map before terrain analysis; (b) map after identification of key locations; (c) map after determination of restricted zones; (d) map after determination of the strategic zones by a progressive expansion of key locations and fusion of adjacent zones.

So, a state representation composed of 64 variables arranged in two main datasets is built for the higher level of the hierarchy: 20 variables for group description and 44 variables for strategic zone description (2 description variables x 11 strategic zones x 2 players). The river is excluded of the representation because it is inaccessible to units.

In addition, high-level maneuvers are associated with key locations on the map to compose an abstracted action space of 49 strategic actions (4 high-level maneuvers x 12 key locations + *Wait order*).

5.2.3 Generalization of Value Functions

The state and action spaces obtained through our abstraction algorithm seem more accessible to state-of-the-art RL techniques, though the number of possible states is still large. In order to generalize between similar situations, we tested both a 3-layer artificial neural network setup trained by the error back-propagation algorithm, and a CMAC setup with unsafe hashing (collisions are ignored).

For the 3-layer artificial neural network setup, we use two fully-connected cascade feed-forward architectures with 80-hidden neurons:

- An unique neural network composed of m inputs $v_1,v_2,\ldots,v_m$ corresponding to the variables of the state representation $s=(v_1,v_2,\ldots,v_m)$ and n outputs $Q(s,a_1),Q(s,a_2),\ldots,Q(s,a_n)$ corresponding to the cumulative rewards associated with the action space $A=\{a_1,a_2,\ldots,a_n\}$ (see Figure 17a);
- A specific neural network for each action a_i of the action space $A=\{a_1,a_2,\ldots,a_n\}$, composed of m inputs $v_1,v_2,\ldots,v_m$ corresponding to the variables of the state representation $s=(v_1,v_2,\ldots,v_m)$ and 1 output $Q(s,a_i)$ corresponding to the cumulative reward associated with the action a_i (see Figure 17b).

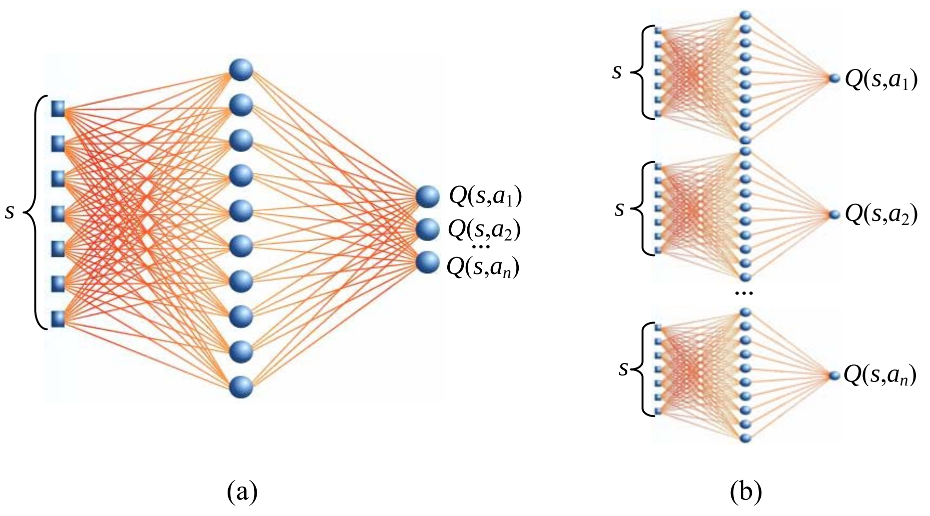

Figure 17. Architectures for the 3-layer artificial neural network setup: (a) a unique neural network for all actions; (b) a specific neural network for each action.

They are tuned according to (LeCun et al., 1998). The hidden layer propagates values by a hyperbolic tangent function and uses a learning rate $\alpha_{hidden} = \alpha_{output} / \sqrt{out}$, where *out* is the number of neurons of the output layer. The output layer propagates values by an ordinary linear function. Connection weights are initialized random uniform to $[-\sqrt{c}\ldots+\sqrt{c}]$, where *c* is the number of connections feeding into the node.

For the CMAC setup, we use a specific CMAC for each action a_i of the action space $A=\{a_1,a_2,\ldots,a_n\}$, composed of *m* inputs $v_1,v_2,\ldots,v_m$ corresponding to the variables of the state representation $s=(v_1,v_2,\ldots,v_m)$ and 1 output $Q(s,a_i)$ corresponding to the cumulative reward associated with the action a_i (see Figure 18). They are tuned according to (Santamaria, Sutton & Ram, 1998). Each CMAC contains 32 overlapping tilings in which the variables $v_1,v_2,\ldots,v_m$ are partitioned into 20 simple tiles.

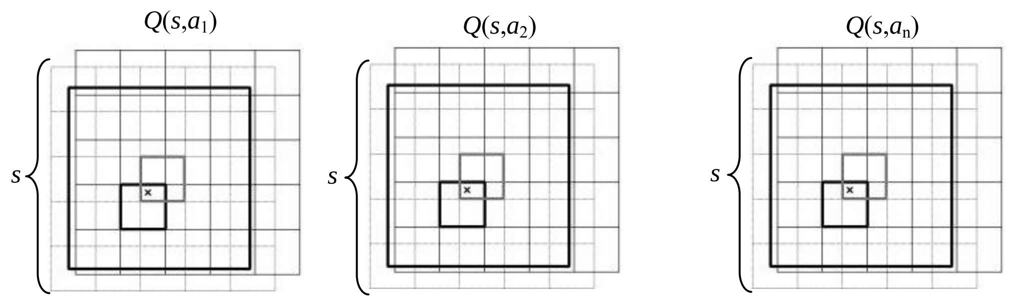

Figure 18. Architectures for the CMAC setup: a specific CMAC for each action.

For all architectures, variables are normalized and centered to [–0,1…+0,1]. Cumulative rewards of selected actions are updated incrementally at each game turn *t* by the gradient descent Sarsa(λ) algorithm (Sutton & Barto, 1998):

$$\vec{\theta}_{t+1} = \vec{\theta}_t + \alpha\left[r_{t+1} + Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)\right]\vec{e}_t,$$

where $\vec{\theta}$ is the weight vector for our function approximator architecture, α is the learning rate and $\vec{e}$ the eligibility trace:

$$\vec{e}_t = \lambda\vec{e}_{t-1} + \nabla_{\vec{\theta}_t} Q(s_t,a_t)$$

with $\vec{e}_0 = \vec{0}$.

Sarsa(λ) is tuned so as to use a learning rate α=0.05 decaying by 10% at every evaluation. The effect of varying the decay rate λ for the eligibility trace is not yet clear, so here we only use λ=0.7 which was the best configuration in past experiments. The action selection is carried out by using the ε-greedy strategy (Kaelbling, Littman & Moore, 1996) with a constant exploration rate ε=0.01 which appears sufficiently exploratory without significantly affecting final performance.

5.2.4 Learning Scenario

The learning scenario is configured according to the STRADA bootstrap mechanism. This mechanism generates valuable experience and leaves only a small part of the hierarchical decision-making to be learned at any given time. So, we configure on one hand, our learning system (called learning AI) to control only the decision-making at the higher level of the French army, giving strategic orders to its subordinate level (3 leaders are present in the higher level of the game scenarios used here). On the other hand, we use the decision-making system included in Battleground™ (called bootstrap AI) to control the Russian army and the lower level of the French army by following the orders flowing down from the level controlled by the learning AI (see Figure 19).

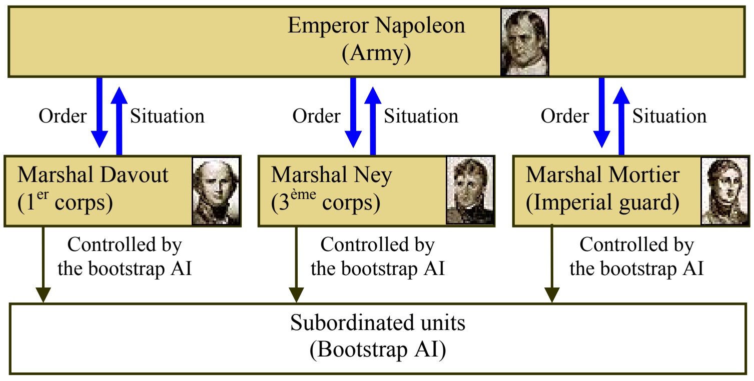

Figure 19. Decision-making scheme for the side of the learning agents.

We also investigate the situation where roles are reversed, learning a Russian strategy against the French. This is relevant as the battle is not symmetrical, since the French army is attacking and the Russian one is defending. Moreover, four hypotheses are tested for our learning agents:

- 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 given to same-level leaders in the previous game turn. For this purpose, we need to take into account three variables which describe an order (a high-level maneuver combined with a key location – axis x and y) for each partner of the same level. Considering both scenarios are composed of three army corps, we add two orders (6 variables) to the state representation of each leader. This generates a state representation of 38 (32 + 6) variables for the *Death in the Flèches* scenario and of 70 (64 + 6) variables for *The Shevardino Redoubt* scenario;
- How best can each leader be rewarded for its order? Two simple options follow: 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 the group, as well as group and opponent losses). 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 by:

$$score_t = \sum_{k=0}^{t}(conqueredObjectives_k - lostObjectives_k + opponentUnitLess_k - ownUnitLess_k)\,.$$

The criterion *R* optimized by RL is then:

$$R = r_1 + r_2 + r_3 + \cdots + r_T = \sum_{t=0}^{T-1} r_{t+1}$$

$$R = \sum_{t=0}^{T-1} score_{t+1} - score_t \equiv score_T - \cancelto{0}{score_0} = score_T\,;$$

- Can self-play approach work as well as when using an existing opponent to play against in order to learn a behavioral strategy to the high level of the hierarchy? This could be another direction to take in order to facilitate the conception of a strategic decision-making system for modern strategy games because one needs just a tactical decision-making system to control the lower level of all sides;
- Can some form of dynamic exclusion of key locations in the action selection process improve performance? For instance, when there are no units in a zone, is it relevant to select maneuvers towards key locations placed in this zone? So, we apply two simple conditions to restrict the action selection process here: (1) a leader only can give attacking orders towards key locations that are not dominated by allied units; and (2) a leader only can give defending orders towards key locations that are not dominated by enemy units.

We compare the performance of several setups of our learning agents with the ones obtained by a random agent, the bootstrap AI itself and a human player (the author of this thesis). All setups (the human player included) are evaluated using the same configuration: they control the decision-making of the leaders of the higher level of hierarchy. They play always against the same opponent: the commercial decision-making system (the bootstrap AI), except for the self-play setup of the learning agents. They use all the tactical decision-making system of the bootstrap AI to control their subordinates.

The performances of the random agent, the bootstrap AI and the human player are evaluated on a unique set of 50 games (or episodes). The learning agents are trained for 10000 learning episodes by alternating learning and evaluation phases of 250 and 50 episodes respectively. During the evaluation phases, both learning and exploration are disabled.

6 Experiments with Battleground™

In this section, we present experiments using our approach applied to Battleground™ to evaluate the performances of the whole system. In order to carry out the experiments, we use the *Death in* *the Flèches* and *The Shevardino Redoubt* scenarios and learn behavioral strategies for French and Russian armies. Although we have tested all function approximator setups presented in the Section 5.2.3, we only show here the results achieved with a 3-layer artificial neural network architecture: an unique network for all actions.

6.1 Death in the Flèches Scenario

6.1.1 Learning Strategies for the French Army (Attacker)

In this first experiment, our agents learn a behavioral strategy for the French army. Their goal is to maximize the score. Figure 20 shows the performances of seven setups for the learning agents:

- Four setups with and without communication combined with global and local rewards. The learning agents have as opponent the bootstrap AI;
- Two setups without communication combined with global and local rewards. The learning agents have as opponents other learning agents (self-play);
- A setup without communication combined with global rewards. Dynamic exclusion of key locations is applied in the action selection process. The learning agents have as opponents the bootstrap AI.

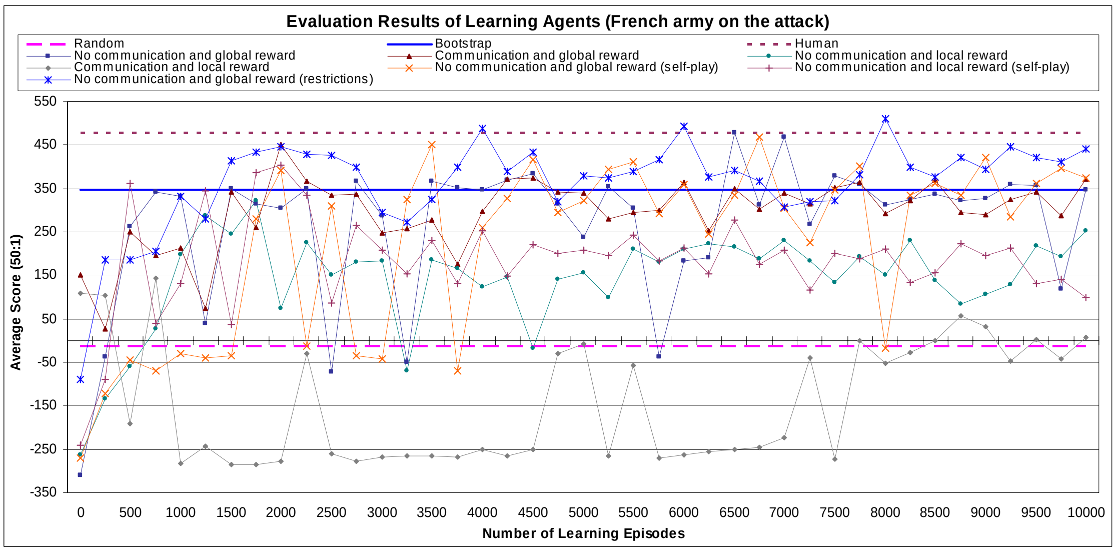

Figure 20. Results for the French army comparing learning agents with and without communication, with local and global reward, playing against the bootstrap AI opponent and by self-play, and baselines (random, bootstrap AI, human).

The performances are 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 performances as the bootstrap AI in only a few thousands of episodes. They correspond to the ones using a global reward. 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 21);
- Communication of orders between agents has little impact on the convergence point, but seems to help to get a smoother progression of performances;
- A self-play setup also converges towards the level of the bootstrap AI, although it takes more time to do it;
- Performances increase a lot when key locations are excluded dynamically in the action selection. This setup manages in some evaluations to approach closely the performances of the human player;
- Local rewards do not lead to good results in these experiments. They lead to an average performance (somewhere between random and bootstrap AI) if no communication was present and very poor one if there is communication. The strategy learned by this setup shows that the agents do not cooperate in the execution of their tasks. The 3rd corps prefers to beat a retreat instead of helping the 1st corps in the combat against Russians since it undergoes heavy losses and never conquer scenario objectives. Consequently, the 1st corps alone becomes hopeless against the strength and the privileged location of the Russian army.

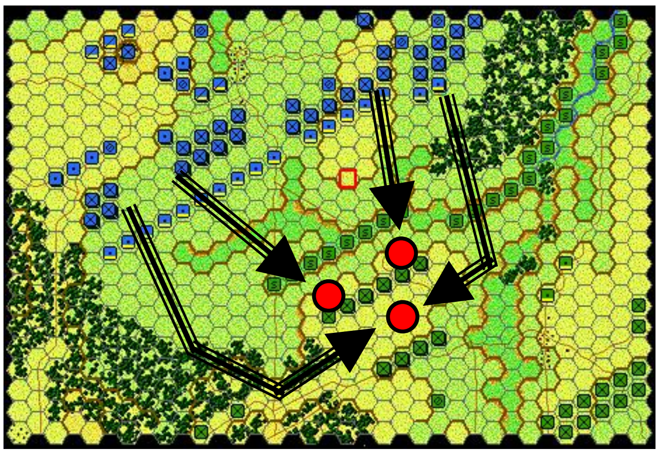

Figure 21. General outline of the behavioral strategy learned for the French army in the *Death in the Flèches* scenario.

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. The other ones could eventually overcome the defenders, a behavior which would be naturally discouraged by a local (selfish) reward.

6.1.2 Learning Strategies for the Russian Army (Defender)

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 22 shows the performances of six setups for the learning agents:

- Four setups with and without communication combined with global and local rewards. The learning agents have as opponent the bootstrap AI;
- Two setups without communication combined with global and local rewards. The learning agents have as opponents other learning agents (self-play).

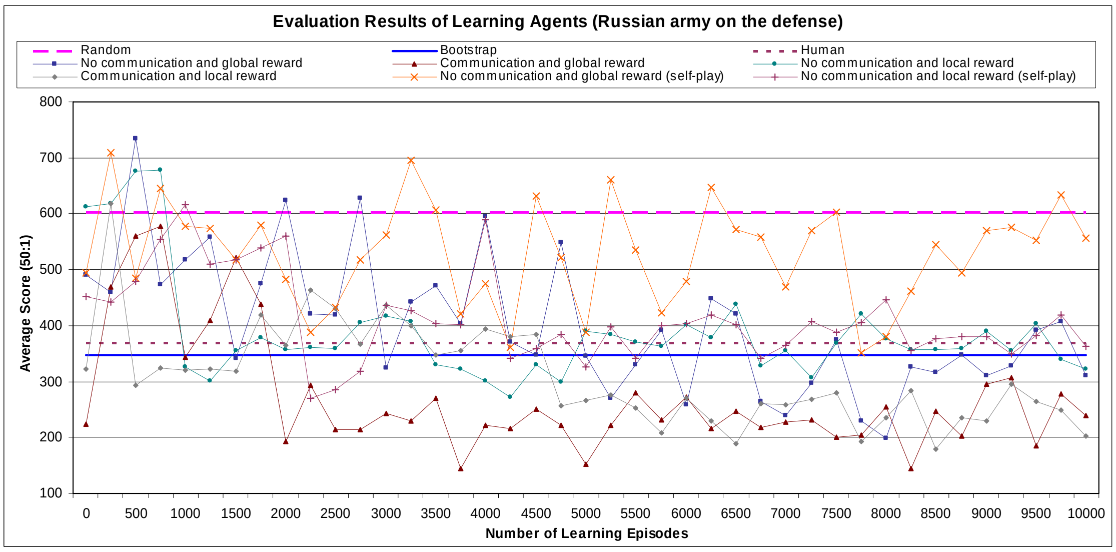

Figure 22. Results for the Russian army comparing learning agents with and without communication, with local and global reward, playing against the bootstrap AI opponent and by self-play, and baselines (random, bootstrap AI, human). *Lowest score is best here.*

We observe the following results:

- All setups obtain interesting performances (at least the same level of the bootstrap AI and the human player) with more stable curves than the attacker side, except when applying the self-play approach combined with global rewards. Indeed because defending strategies seem simpler as a result of the relatively low mobility of the simulated armies;
- Communication of orders between agents increases performances significantly, help also them to get a smoother progression. Setups based on communication achieve excellent results, outperforming by far the bootstrap AI and the human player;
- Contrarily of the attacker side, the self-play setups do not achieve here the same performances of the learning agent when playing against the bootstrap AI;
- 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.

6.2 The Shevardino Redoubt Scenario

6.2.1 Learning Strategies for the French Army (Attacker)

In this third experiment, our agents learn a behavioral strategy for the French army placed on *The Shevardino Redoubt* scenario. Their goal is to maximize the score. Figure 23 shows the performances of five setups for the learning agents:

- Four setups with and without communication combined with global and local rewards. The learning agents have as opponent the bootstrap AI;
- A setup without communication combined with global rewards. Dynamic exclusion of key locations is applied in the action selection process. The learning agents have as opponents the bootstrap AI.

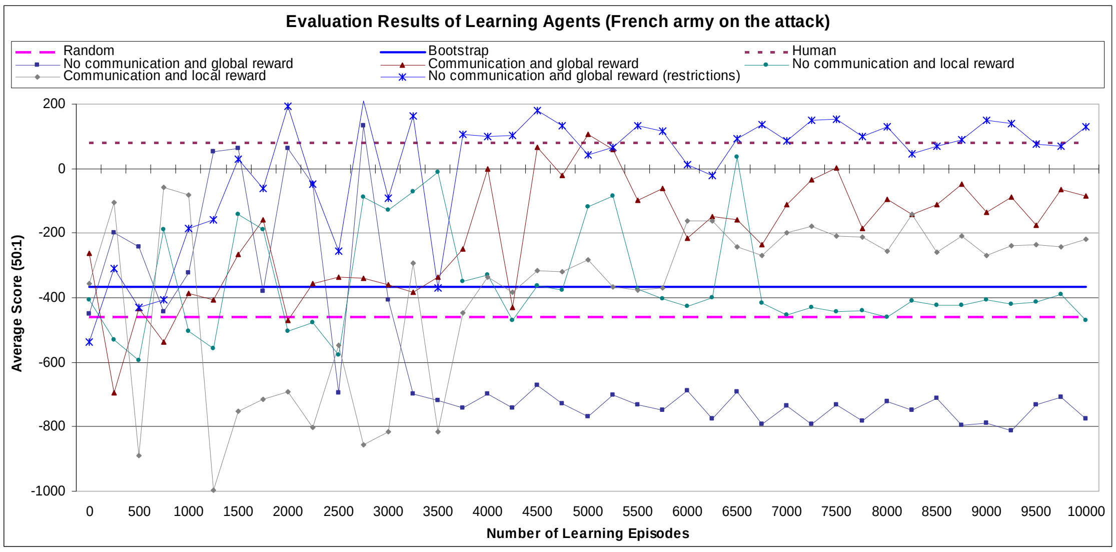

Figure 23. Results for the French army comparing learning agents with and without communication, with local and global reward, and baselines (random, bootstrap AI, human).

We observe the following results:

- Several setups largely exceeded the very poor performances of the bootstrap AI. They correspond to the ones communicating orders between agents or excluding dynamically key locations. These results are very encouraging because *The Shevardino Redoubt* scenario requires, *a priori*, a more refined strategy than *Death in the Flèches*;
- Communication of orders between agents has a positive impact on the convergence point;
- Once again, performances increase a lot when key locations are excluded dynamically in the action selection. This setup surpasses the performances of the human player on several evaluations. Its strategy consists in threatening the enemy by encompassing him on his flanks, leaving him with only alternative of attacking or abandoning his elevated position without fighting (see Figure 24). *The Shevardino Redoubt* scenario comprises a map made up of several hills, forests and a river. It is well known that hills, forests, and rivers are

difficult obstacles to the march of an army. Among hills, a great number of locations are found to be very strong in themselves, and dangerous to attack. In hill combat, the attacker has always the disadvantage. This poses a particular problem in this scenario because the most important objective is placed at the top of a hill completely dominated by the Russian army at the beginning of the game. So, the key initiative is to threat the opponent without engaging a frontal attack. This corresponds precisely to the strategy that our system has learned.

- The setup without communication combined with global rewards obtained interesting performances on the first three thousands episodes. However, it abruptly changed its strategy and declined to very poor performances. This is derived from a strategy which consists in massively attacking the opponent, causing enormous losses and an absolute defeat. This strategy is equivalent to that adopted by the bootstrap AI. Curiously, this setup when combined with dynamically exclusion of key locations is the best one in our experiments;

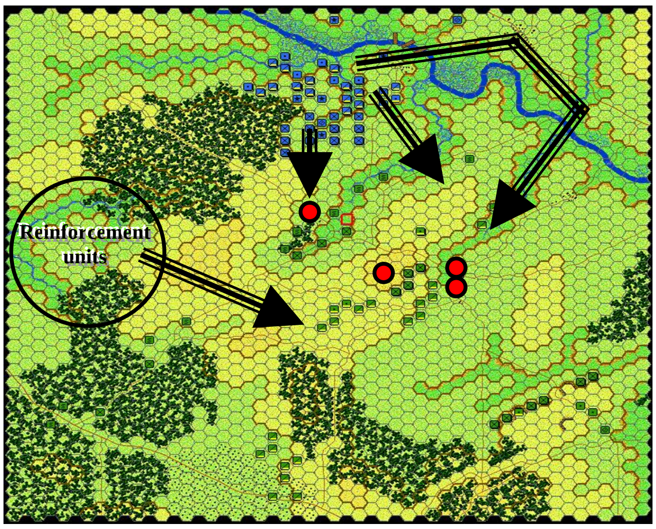

Figure 24. General outline of the behavioral strategy learned for the French army in *The Shevardino Redoubt* scenario.

6.2.2 Learning Strategies for the Russian Army (Defender)

In this fourth experiment, we reverse once again the roles by asking our agents to learn a strategy for the Russian army. Their goal is then *to minimize the game score*. Figure 25 shows the performances 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.

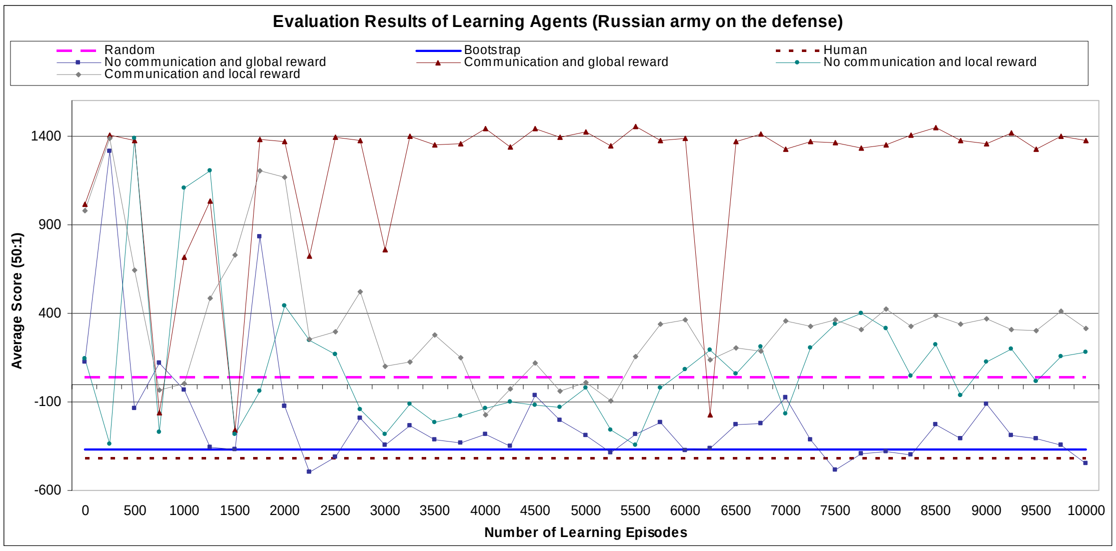

Figure 25. Results for the Russian army comparing learning agents with and without communication, with local and global reward, and baselines (random, bootstrap AI, human). *Lowest score is best here.*

We observe the following results:

- In contrast with the attacker agents, the best performances of the defenders are those obtained without communication. They achieve performances very similar to the bootstrap AI and the human player. Since strategic decision-making for the attacker side is more difficult than for the defender side in this scenario, the fact that the defensive agents obtained similar performances the bootstrap AI is a very good result. Indeed, the defender side is placed on a privileged location and movement becomes hard in the second half of the game because it takes place after dark. This is why the strategy adopted by the bootstrap AI and by the learning agents is very successful. It consists in keeping the privileged location at the top of the hill;
- Unexpectedly, the setup with communication and global reward obtains very poor performances in this experiment. This setup had achieved until now the best performances in all other experiments. Its strategy consists in moving units back, going out from the privileged location at the top of the hill in order to escape of the French army attacks. So, French units conquer easily the most important scenario objective, causing a very bad score for Russians;

6.3 Conclusion

We evaluated the STRADA approach with Battleground™ by using a historical military hierarchy of command and control. The goal was to learn a high level behavioral strategy for the French (attacker) and Russian (defender) armies.

We studied several setups for the learning agents by combining coordination and rewarding strategies in the framework of the *Death in the Flèches* and *The Shevardino Redoubt* scenarios.

Moreover, we tested the self-play approach to simultaneously learn a behavioral strategy for attacker and defender.

The results obtained are very confident since our decision-making system largely exceeded, in both scenarios, the performances of the existing Battleground™ AI. In this context, 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 performances of a system by learning only a part of a global strategy.

7 Conclusion and Future Work

7.1 Summary and contributions

Modern strategy games present a very complex and original problem little investigated until now. Designing efficient decision-making systems for this type of games raises difficult questions to the AI community and interests directly the game industry.

In this thesis, we proposed a novel integrated learning approach to the automatic design of adaptive behavioral strategies for modern strategy games. Our approach, called STRADA, is an alternative to techniques generally employed by the industry. It combines state-of-the-art techniques from several areas of machine learning with new ideas to deal with the complexity of these games. In particular, it investigates two main issues: (1) complexity reduction of the problem by decomposing the decision-making and abstracting state and action spaces; and (2) acceleration of the process of learning from interaction by generalizing the value function and bootstrapping the acquisition of experience.

The decision-making based on a hierarchical organization of agents seems natural in modern strategy games. This organization allows simultaneously to: (1) guide the abstraction of state and action spaces to design relevant representations to each type of agent, (2) limit the communication between agents, (3) organize value functions, and (4) produce a mechanism of learning by stages.

The design of state and action representations is a crucial stage for the use of learning techniques. For this purpose, we designed a terrain analysis algorithm which exploits the essential properties of a geographical map. This algorithm reduces the huge size of state and action spaces of modern strategy games to much more reasonable sizes. It is a significant accomplishment because most applications applying reinforcement learning have generally used carefully hand-crafted domain features. The generated representations are applied to function approximators in order to generalize the value function and, consequently, the behavioral strategy learned by our system.

Moreover, we accelerated the process of learning from interaction by setting up a particular mechanism for the acquisition of experience. This mechanism leads the learning agents to learn progressively a behavioral strategy by treating only a part of the hierarchical structure in each step. For this purpose, an existing tactical decision-making system is employed as support.

We evaluated the approach on the task of learning valuable behavioral strategies for a commercial wargame, Battleground™. The results obtained are very significant since they outperformed by far the existing commercial script-based solution, despite the limitation resulting from controlling only the high level of the hierarchy.

7.2 Future Work

7.2.1 Abstraction

STRADA actually requires variables that must be used as a summary of the state of the groups of units. Although we gave directions in order to find relevant variables here and obtained good results with them, ideally they would be found automatically in order to fully automate our abstraction procedure. For this purpose, we mention some work which can guide this automation (Giunchiglia & Walsh, 1992; Blum & Langlay, 1997; Saitta & Zucker, 2001; Jong & Stone, 2005; Li, Walsh & Littman, 2006).

Moreover, high-level behaviors used to abstract elementary tasks must be also provided to STRADA. Some examples such as monitoring, exploration and occupation of the terrain were mentioned and are natural here. Nevertheless, we believe that by exploring static and dynamic terrain information, the design of these behaviors can be considerably helped and eventually automated. This is why techniques of terrain analysis and qualitative spatial reasoning should once again be considered. A brief study in this context was made on premises of this thesis (Corruble, Madeira & Ramalho, 2002). A follow-up study would be useful.

Furthermore, strategies learned with STRADA are specific to a given scenario. This is why state representation contains specific information to a given map (static zones generated by our terrain analysis algorithm). It would be interesting to try alternatives taking into account dynamic zones in order to generate more general representations. This point is certainly another direction to be taken in order to bring the strategy learned by our learning agents adaptable to different and dynamic maps. Recent work tackled this problem (Guestrin et al., 2003a). However, this was only applied to very simple scenarios.

7.2.2 Learning and coordination between agents

The bootstrap mechanism proposed by STRADA allows progressive learning. However, the experiments we carried out designed strategies for the highest levels of the hierarchy where we began to tackle the issues of rewarding and coordination between agents. To control all the levels of the strategic decision-making, it becomes crucial to look further into the study of these questions which pose an interesting challenge field of research (Claus & Boutilier, 1998; Guestrin, Lagoudakis & Parr, 2002; Chalkiadakis & Boutilier, 2003; Pynadath & Tambe, 2002).

In addition, this bootstrap mechanism can generate overspecialized strategies and the learning process might suffer from shortcomings (high-level strategies can be unsuccessful either because they are inherently flawed, or because they have not been properly implemented at the lower level). Let us note though that this limitation could possibly be overcome by looping the bootstrap mechanism, bringing agents to learn by self-play. However, we did not have time to test this idea in this thesis.

Finally, the decision-making of STRADA agents was formalized without taking into account the non-Markovian context of multi-agent environment in modern strategy games. The environment becomes partially observable as soon as each agent has its own abstracted state representation. Moreover, we did not treat here the “fog of war” technique which is generally used in modern strategy games to increase realism by hiding from the players all that is not in the field of vision of their units. These problems can open several research directions in the domains of games and reinforcement learning. In addition, our case study was a turn-based game which seems well adapted to the sequential framework of reinforcement learning. In the future, STRADA should also be tested with a real-time strategy game in order to be evaluated in this case and eventually adapted if it is needed.

7.3 Last Word

We proved in this thesis that it is possible to use reinforcement learning techniques to design adaptive behavioral strategies for modern strategy games. In addition to that, we believe the ideas and concepts presented here are not specific to games, but they could be also applied to most large-scale multi-agent simulations.

Since very sophisticated real-world environments are simulated by modern video games, we have much to learn from the study of them. These environments can be used as a laboratory to validate new AI techniques. They allow to ensure that the techniques are well adapted to the real world and consequently can open a path to their application to current situations of the real life.

References

David Andre and Stuart Russell. State abstraction for programmable reinforcement learning agents. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pp. 119-125, AAAI Press, 2002.

Leemon C. Baird and A. Harry Klopf. Reinforcement Learning with High-Dimensional, Continuous Actions. Technical Report WL-TR-93-1147 Wright Laboratory, Wright-Patterson Air Force Base, 1993.

Andrew G. Barto and Sridhar Mahadevan. Recents Advances in Hierarchical Reinforcement Learning. *Discrete-Event System Journal*, 2003.

Avrim L. Blum and Pat Langley. Selection of Relevant Features and Examples in Machine Learning. *Artificial Intelligence*, 97(1-2):245-271, 1997.

Craig Boutilier. Planning, Leaning and Coordination in Multiagent Decision Processes. In *Proceedings of the Sixth Conference on Theoretical Aspects of Rationality and Knowledge*, pp.195-210, The Netherlands, 1996.

Craig Boutilier, Richard Dearden and Moises Goldszmidt. Stochastic Dynamic Programming with Factored Representations. *Artificial Intelligence*, 121(1-2):49-107, 2000.

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

Anthony R. Cassandra. Exact and Approximate Algorithms for Partially Observable Markov Decision Processes. *Ph.D. Thesis*. Brown University, Department of Computer Science, 1998.

Georgios Chalkiadakis and Craig Boutilier. Coordination in multiagent reinforcement learning: A Bayesian approach. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pp.709-716, 2003.

Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pp.746-752, 1998.

Anthony G. Cohn and Shyamanta M. Hazarika. Qualitative Spatial Representation and Reasoning : An Overview. *Fundamenta Informaticae*. 46(1-2):1-29, 2001.

Vincent Corruble. AI approaches to developing strategies for wargame type simulations. In *Proceedings of the AAAI Fall Symposium on Simulating Human Agents*. Cape Cod, USA, 2000.

Vincent Corruble, Charles Madeira and Geber Ramalho. Steps Toward Building a Good AI For Complex Wargame-Type Simulation Games. In *Proceedings of the Third International Conference on Intelligent Games and Simulation*, London, UK, 2002.

Rémi Coulom. *Reinforcement Learning Using Neural Networks, with Applications to Motor Control*. PhD thesis, Institut National Polytechnique de Grenoble, Grenoble, France, 2002.

Robert H. Crites and Andrew G. Barto. Improving Elevator Performance Using Reinforcement Learning. *Advances in Neural Information Processing Systems*, 8:1017-1023, MIT Press, Cambridge, MA, 1996.

Robert H. Crites and Andrew G. Barto. Elevator group control using multiple reinforcement learning agents. *Machine Learning*, 33:235-262, 1998.

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

Alain Dutech, Olivier Buffet and François Charpillet. Multi-Agent Systems by Incremental Gradient Reinforcement Learning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, Seattle, USA, 2001.

Kenneth D. Forbus, James V. Mahoney and Kevin Dill. How qualitative spatial reasoning can improve strategy game AIs. In *Proceedings of AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment*, Stanford, CA, 2001.

Mohammad Ghavamzadeh, Sridhar Mahadevan and Rajbala Makar. Hierarchical Multiagent Reinforcement Learning. *Journal of Autonomous Agents and Multiagent Systems*, 13(2):197-229, 2006.

Fausto Giunchiglia and Toby Walsh. A Theory of Abstraction. *Artificial Intelligence*, 56(2-3):323-390, 1992.

Charles Grindle, Michael Lewis, Robin Glinton, Joseph Giampapa, Sean Owens and Katia Sycara. Automating Terrain Analysis: Algorithms for Intelligence Preparation of the Battlefield. In *Proceedings of the Human Factors and Ergonomics Society*, Santa Monica, CA, 2004.

Carlos Guestrin, Daphne Koller, Chris Gearhart and Neal Kanodia. Generalizing Plans to New Environments in Relational MDPs. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, 2003a.

Carlos Guestrin, Daphne Koller and Ronald Parr. Multiagent Planning with Factored MDPs. In *Advances in Neural Information Processing Systems*, pp.1523-1530, 2001.

Carlos Guestrin, Daphne Koller, Ronald Parr and Shobha Venkataraman. Efficient Solution Algorithms for Factored MDPs. *Journal of Artificial Intelligence Research*, 19:399-468, 2003b.

Carlos Guestrin, Michail Lagoudakis and Ronald Parr. Coordinated Reinforcement Learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 227-234, Menlo Park, Calif.: AAAI Press, 2002.

Junling Hu and Michael P. Wellman. Multiagent Reinforcement Learning: Theoretical Framework and Algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 242-250, Morgan Kaufman, San Francisco, CA, 1998.

Junling Hu and Michael P. Wellman. Nash Q-learning for General-Sum Stochastic Games. *Journal of Machine Learning Research*, 4:1039-1069, 2003.

Nicholas K. Jong and Peter Stone. State Abstraction Discovery from Irrelevant State Variables. In *Proceedings of Nineteenth International Joint Conference on Artificial Intelligence*, 2005.

Leslie P. Kaelbling, Michael L. Littman and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99-134, 1998.

Leslie P. Kaelbling, Michael L. Littman and Andrew W. Moore. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237-285, 1996.

Spiros Kapetanakis and Daniel Kudenko. Reinforcement Learning of Coordination in Cooperative Multi-agent Systems. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pp.326-331, Edmonton, Alberta, Canada, 2002.

Spiros Kapetanakis and Daniel Kudenko. Reinforcement Learning of Coordination in Heterogeneous Cooperative Multi-agent Systems. In *Proceedings of the Fourth AISB Symposium on Adaptive Agents and Multi-agent Systems*, 2004.

Yann LeCun, Leon Bottou, Genevieve B. Orr and Klaus-Robert Müller. Efficient BackProp. *Neural Network: Tricks of the Trade*, 1524(1), Springer-Verlag, 1998.

Lihong Li, Thomas J. Walsh and Michael L. Littman. Towards a Unified Theory of State Abstraction for MDPs. In *Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics*, pp.531-539, 2006.

Michael L. Littman. Markov Games as a Framework for Multi-Agent Reinforcement Learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 157-163, Morgan Kaufmann, New Brunswick, NJ, 1994.

Michael L. Littman. Friend-or-Foe Q-learning in General-Sum Games. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 322-328, Morgan Kaufmann, San Francisco, CA, 2001.

Charles Madeira, Vincent Corruble, Geber Ramalho and Bohdana Ratitch. 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, San Jose, CA, 2004.

Charles Madeira, Vincent Corruble and Geber Ramalho. Generating Adequate Representations for Learning from Interaction in Complex Multiagent Simulations. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, France, 2005.

Charles Madeira, Vincent Corruble and Geber Ramalho. Designing a Reinforcement Learning-based Adaptive AI for Large-Scale Strategy Games. In *Proceedings of the Second Conference on Artificial Intelligence and Interactive Digital Entertainment*, Marina del Rey, CA, 2006.

Rajbala Makar, Sridhar Mahadevan and Mohammad Ghavamzadeh. Hierarchical Multi-Agent Reinforcement Learning. In *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 246-253, ACM Press, Canada, 2001.

Thomas Malone and Kevin Crowston. The Interdisciplinary Study of Coordination. *ACM Computing Surveys*, 26(1):87-119, 1994.

Bhaskara Marthi, Stuart Russell, David Latham and Carlos Guestrin. Concurrent Hierarchical Reinforcement Learning. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, Edinburgh, Scotland, 2005.

Alexander Nareyek. AI in Computer Games. *ACM Queue*, 1, 2004.

Sebastien Paquet. Distributed Decision-Making and Task Coordination in Dynamic, Uncertain and Real-Time Multiagent Environments. *Ph.D. thesis*, Université Laval, 2006.

Ronald Parr and Stuart Russell. Reinforcement Learning with Hierarchies of Machines. *Advances in Neural Information Processing Systems*, 10, MIT Press, 1998.

David Pynadath and Milind Tambe. The Communicative Multiagent Team Decision Problem: Analyzing Teamwork Theories and Models. *Journal of Artificial Intelligence Research*, 2002.

Steve Rabin. *AI Game Programming Wisdom 2*. Charles River Media, 2003.

Bohdana Ratitch and Doina Precup. Characterizing Markov Decision Process. In *Proceedings of the Thirteenth European Conference on Machine Learning*, Finland, 2002.

Bohdana Ratitch and Doina Precup. Sparse Distributed Memories for On-Line Value-Based Reinforcement Learning. In *Proceedings of the Fifteenth European Conference on Machine Learning*, pages 347-358, Pisa, Italy, 2004.

Bohdana Ratitch. On Characteristics of Markov Decision Process and Reinforcement Learning in Large Domains. *Ph.D. Thesis*, School of Computer Science, McGill University, 2005.

Martin Riedmiller and Artur Merke. Using machine learning techniques in complex multi-agent domains. In *I. Stamatescu, W. Menzel, and U. Ratsch, editors, Perspectives on Adaptivity and Learning*, Springer, 2002.

Lorenza Saitta and Jean-Daniel Zucker. A Model of Abstraction in Visual Perception. *Applied Artificial Intelligence*, 15(8):761-776, 2001.

Juan C. Santamaria, Richard S. Sutton and Ashwin Ram. Experiments with Reinforcement Learning in Problems with Continuous State and Action Spaces. *Adaptive Behavior*, 6(2):163-218, 1998.

Jonathan Schaeffer and H. Jaap Van den Herik. Games, computers, and artificial intelligence. *Artificial Intelligence*, 134(1-2):1-8, 2002.

Sandip Sen and Gerhard Weiss. Learning in Multiagent Systems. In *Weiss G., editor, Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pp.259-298. MIT Press, 2000.

Herbert A. Simon. *The Sciences of the Artificial*. MIT Press, Cambridge, MA, 1981.

William D. Smart and Leslie P. Kaelbling. Practical Reinforcement Learning in Continuous Spaces. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 903-910, Morgan Kaufmann, San Francisco, CA, 2000.

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

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning, An Introduction*. MIT Press, Cambridge, MA, 1998.

Richard S. Sutton, Doina Precup and Satinder Singh. Between MDPs and semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artificial Intelligence*, 112:181-211, 1999.

Milind Tambe and Weixiong Zhang. Towards Flexible Teamwork in Persistent Teams: Extended Report. *Journal of Autonomous Agents and Multi-Agent Systems*, 3(2):159-183, Kluwer Academic Publishers, Hingham, MA, 2000.

Gerald Tesauro. Programming backgammon using self-teaching neural nets. *Artificial Intelligence*, 134:181-199, 2002.

Gerald Tesauro. Extending Q-Learning to General Adaptive Multi-Agent Systems. *Advances in Neural Information Processing Systems*, 16:871-878, MIT Press, Cambridge, MA, 2004.

Sebastien Thrun and Anton Schwartz. Issues in Using Function Approximation for Reinforcement Learning. In *Proceedings of the Fourth Connectionist Models Summer School*, Hillsdale, NJ, 1993.

Kagan Tumer and David Wolpert. A Survey of Collective Intelligence. In *Tumer K., and Wolpert D. (eds)* Collectives and the Design of Complex Systems, Springer-Verlag, 2004.

William Uther and Manuela Veloso. Adversarial Reinforcement Learning. *Technical Report*. Computer Science Department, Carnegie Mellon University, 1997.

Chris Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8:279-292, 1992.

Gerhard Weiss. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 2000.

Shimon Whiteson and Peter Stone. Concurrent Layered Learning. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 193-200, Melbourne, Australia, 2003.

David Wolpert and Kagan Tumer. An Introduction to Collective Intelligence. *Technical Report NASA-ARC-IC-99-63*, NASA Ames Research Center, 1999.