Bootstrapping the Learning Process for the Semi-automated Design of a Challenging Game AI

Charles Madeira[1], Vincent Corruble[1], Geber Ramalho[2], Bohdana Ratitch[3]

[1]Laboratoire d'Informatique de Paris 6
Université Pierre et Marie Curie (Paris 6)
4 Place Jussieu
75252 Paris Cedex 05 FRANCE
{Charles.Madeira,Vincent.Corruble}@lip6.fr

[2]Centro de Informática
Universidade Federal do Pernambuco
Caixa Postal 7851, Cidade Universitária
50732-970 Recife, PE, BRAZIL
glr@cin.ufpe.br

[3]School of Computer Science
McGill University
3480 University St., Office 112
Montreal, H3A 2A7 CANADA
bohdana@cs.mcgill.ca

Abstract

This paper proposes a methodology for the semi-automated design of a game AI for simulation and strategy games which require the player to control a potentially high number of characters or units in complex environments. After defending the idea of using Machine Learning, and especially Reinforcement Learning, as a basic technique, some of the key issues, mostly dealing with complexity, representation, and coordination, are outlined, and some ways forward are proposed. These revolve mainly around the idea of problem decomposition using the game structure, and of bootstrapping the learning process by letting the learning game AI play against another AI, and only progressively take more and more control of the decision-making. The ongoing application of this methodology to the design of a new game AI for an existing wargame system is described in some detail.

Introduction

This paper addresses the issue of the design of a game AI for a category of games which range from real-time action-oriented strategy games, such as Age of Empires (Microsoft), to intricate historical simulations and wargames, such as Sid Meier's Gettysburg (Firaxis) or John Tiller's Battleground series (Talonsoft).

In this paper we focus on a new approach to the semi-automatic generation of a game AI using Machine Learning techniques. We advocate a specific methodology that one can follow to circumvent typical difficulties arising in the context of large scale complex strategic games. We take as an ongoing case study the use of Reinforcement Learning techniques to develop an AI for a particular historical wargame, but we believe that the lessons learned have a wide applicability to most uses of Machine Learning for AI design applied to games where a large number of units or characters have to coordinate between themselves to achieve a long-term common goal.

In the following, we introduce the notion of a Learning AI, describe the type of learning technique we used, introduce the bootstrap mechanism and the general methodology of our approach. Then we describe its application to a wargame, give some initial results and discuss ongoing work.

Challenges for the design of a game AI, the Machine Learning option.

The importance of a challenging game AI

It is well-known in the game industry that a challenging AI is one of the keys to retain a high replay value for a game (Nareyek 2004). In fact, an important role for a game AI is to continuously challenge the user at his/her level, provoking him/her to change tactics or strategies. Despite the recognized importance of AI in games, few detailed attempts for developing methodologies for the design of a challenging game AI are found in the literature (Rabin 2002).

Most games, especially for complex simulations that are the focus of our interest, use some form of rule base as the basic technique supporting their game AI (Rabin 2002). Automated reasoning based on rules and inference engines is one of the oldest and most studied techniques developed within academic AI. It comes in various forms ranging from simple scripts (where actions are either time-triggered, or event-triggered), to proper rule bases (from the simple reactive ones, to more advanced cognitive ones which allow for internal states, goals, and planning). The most often cited benefits of rule-based programming include its expressivity, flexibility and intuitiveness. The disadvantages are also well-known. First, as the knowledge and reasoning that rule bases model become complex, their programming becomes seriously difficult. This is indeed a serious limitation if one is interested in their application to complex simulation games. Second, rule bases were developed originally for expert systems, for which the same situation should usually lead to the same reasoning and conclusion. This is typically untrue in games, where from the strategic point of view, any fixed reasoning is easily

caught by the opponent and its loopholes (there are always some) can soon be exploited (Corruble 2000).

For these reasons, it is perfectly legitimate to consider other alternatives to address the design of a game AI. In the following, we consider those proposed by Machine Learning, which are characterized, when compared to rule bases, by a high adaptability, and less reliance on prior knowledge.

Machine learning as a worthy alternative

Almost as old as the field of automated reasoning, the field of Machine Learning is important in academic AI and has contributed a significant number of techniques, methodologies, and algorithms (Mitchell 1997). One of its key concepts is that knowledge can be the output of an automated learning process that takes experience (data) as an input. The kind of data that can be used as input and the form of knowledge produced during the learning process depend mostly on the learning algorithm chosen, which itself is constrained by the learning task. However, there are some generic questions to tackle when using machine learning for the design of a game AI.

The main issue is that there is no learning without experience. Experience can be gathered in all the situations which constitute learning opportunities. Simulation games are a dream application domain for learning techniques as generating data could not be easier: one just has to play. However, to generate worthy data for learning, one needs a worthy opponent. Moreover, the first option which would be to have our game AI play against a (good) human player is not really viable for complex games as it would require the human player to play thousands (potentially hundreds of thousands) of games against a game AI that would initially be a terrible player.

In this paper we introduce an approach and a methodology that we are now experimenting with on a wargame system in order to address this data acquisition problem. We use some form of learning to design a game AI semi-automatically. But to initiate this learning, we let our system play against another game AI already designed for this game, effectively producing some form of a bootstrap mechanism.

Using Reinforcement Learning to design a game AI

Reinforcement Learning (RL) is a general approach for learning from interaction with a stochastic, unknown environment (Sutton and Barto 1998). The field has contributed both efficient algorithms and solid theoretical results about their performances. The RL approach is particularly suited in the case of strategic games, as it is specifically designed for learning sequential decision-making strategies based on long-term performance criteria.

Markov Decision Processes (MDPs) present one natural way to formulate sequential decision-making tasks. An MDP is defined as a 4-tuple (S,A,P,R), in which S is a finite set of states (the situation of the game at a given time), A is a finite set of actions (what actions or orders the player can choose from), P is a state transition probability function, and R is a reward function. Dynamics of the environment is defined by the transition probability function P: S x A x S $\rightarrow$ [0, 1], where $P^a_{ss'}$ denotes the probability that action a, when executed in state s, transfers the system to state s'. In general, for games, P is determined by the game rules. In fact, RL can be applied even without explicitly knowing P, since it can potentially be used for learning a good strategy directly or by first inferring P from experience. The reward function is defined as a real-valued bounded function R: S x A x $S \rightarrow \Re$, where $R^a_{ss'}$ is the reward of taking action a in state s and observing s' at the next state.

Before going further, it is important to determine which decision-making process we wish to model. Is it the one of a human player controlling a potentially high number of game units (or characters) simultaneously? Is it the decision-making of one of these units taken individually (in which case many such independent processes would have to be modeled concurrently)? Actually, these two options are potentially interesting, yet they both present difficulties. The first allows for some explicit coordination, but soon hits a complexity wall from the point of view of Reinforcement Learning, as the number of potential actions considered by the human player grows exponentially with the number of controlled units. The second, decentralized option, lets each unit learn a strategy independently of other units. This is usually suboptimal since, in most situations, some form of coordination to reach a common goal is either necessary or at least helpful.

Standard tabular RL does not scale well to large state and action spaces (Sutton and Barto 1998). As a consequence of the complexity involved in strategy games, RL cannot obtain satisfactory results in a reasonable time based on representations which are very detailed. Since the number of low-level actions available to each character or unit is huge, one can naturally understand that any reasoning at a tactic or strategic level needs to be tackled at a higher, more abstract level. Therefore, a good AI should have various representation granularities, each one adapted to the task at hand (Corruble, Madeira and Ramalho 2002) (Zucker, Bredèche and Saitta 2002). In the next section, we present a way of decreasing the complexity of the representation through hierarchical problem decomposition and information abstraction at each level of the hierarchy.

Methodology for AI automated design via learning

As previously discussed, when a problem is highly complex, it is not feasible to solve it directly as a whole. A basic method to decrease complexity is to decompose the problem into relatively independent subproblems, which are easier to solve individually. In this section, we present a methodology, through which reinforcement learning techniques can be applied to the semi-automated design of a game AI. We believe that this methodology has a wide applicability in the field of strategy and adventure games.

The following section will present the first steps of its application to a specific wargame system.

Decomposing the problem

The nature of many strategic simulation games makes them natural candidates for a hierarchical decomposition approach and multi-agent distributed reasoning. For instance, within a military hierarchy, a leader is able to give an order to a subordinate and some time later, receives some feedback. This structure enables us to decompose the problem so that the decision-making process of a game AI is carried out on several levels (see Figure 1).

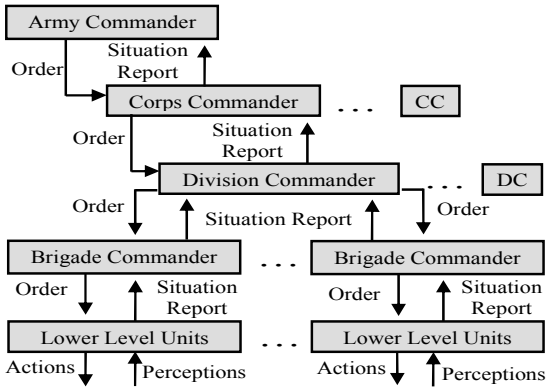

Figure 1: Military Hierarchical Scheme of Command and Control (Multi-agent point of view)

The main advantage of this scheme is that it allows the "higher levels" of the hierarchy to formulate a strategy, without being overwhelmed by the intractably large number of possibilities which the computer AI would have to consider if it possessed all the information pertaining to individual low level units. Consequently, a representation with an appropriate granularity for each level is needed for strategic reasoning, and can be obtained as an abstraction of the low level representation. This is in itself a complex problem. Fortunately, we can use some knowledge about the domain (for instance, military decision-making), a detailed analysis of the rules of the game, or indeed simple common-sense, to guide us toward that goal (Corruble, Madeira and Ramalho 2002).

Bootstrapping the learning process

Once a representation is designed for each level of the hierarchy to be controlled by the game AI, we can tackle the problem of learning a good strategy. As mentioned earlier, we found that this would require the use of a bootstrapping mechanism obtained by playing against another game AI. This mechanism is actually designed along two dimensions: The first type of bootstrap consists in letting initially the learning AI play and learn against another AI (which we call later the *bootstrap AI*). The second type of bootstrap is obtained by letting the learning AI take only a partial control over its camp.

In the first case, the learning AI would take charge of the entire decision making for its camp, while its opponent, rather than a human player, is another AI, produced by other techniques (such as rule bases). This lets our system play thousands of games to learn progressively (and slowly) a better strategy.

Since learning to control an entire camp (at all levels of the military hierarchy) at once remains a very complex task to tackle (Whiteson and Stone 2003), we use also a second type of bootstrap, where the existing non-adaptive bootstrap AI takes partial control of our camp, in addition to the full control of the enemy camp. This leaves only a small part of the decision making process to be learnt at any given time. This is a fundamental point of our methodology because it allows incremental learning of strategies with increasingly refined levels of specificity and detail.

Returning to the example of Figure 1, we could implement this methodology by learning strategies for the highest levels of the military hierarchy first and then progressively descending down. To control our camp, the learning AI chooses high level orders for the commanders it is in charge of, after which the bootstrap AI is used to implement these orders at the lower levels.

Experimental platform

In order to generate valuable experience for our learning AI, we chose to develop a software platform that would serve several functions: (1) high-level management of our experiments, (2) control and parameterization of our learning AI engine, (3) management of communications between our learning AI and the "bootstrap AI". Point (3) is of particular importance because it lets the experimenter manage which part of the decision making is carried out by the learning AI, and which part is left to the bootstrap AI.

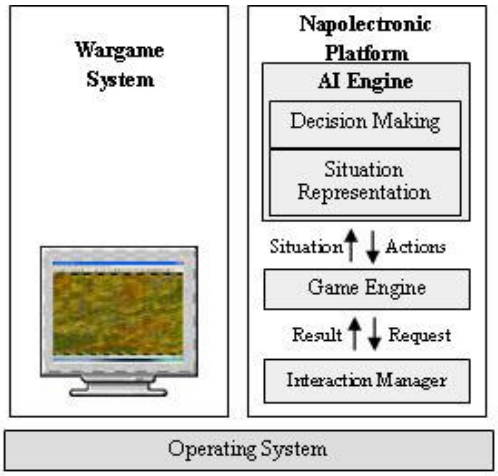

Figure 2: Napolectronic Platform Architecture

Our configurable platform, named *Napolectronic,* has an architecture based on three main modules: a game engine, an AI engine, and an interaction manager with a wargame application (see Figure 2).

The game engine is in high-level control of the game evolution. It takes care of data acquisition and turn/phase control. The AI engine deals with changes of representation, decision making for each unit or character, and learning, as well as all the parameters attached. The interaction manager takes care of managing the interactions between our learning AI and the existing wargame (including its AI we use as "bootstrap").

To conduct our experiments, we chose as specific wargame John Tiller's *Battleground*™ series (Talonsoft). Battleground is a turn-based simulation of Napoleonic battles that aim at a good historical accuracy, with detailed maps, orders of battles, combat resolution, etc. while retaining gameplay value.

Experiments with Battleground

In this section, we present an application of our methodology to the design of a game AI for Battleground.

Figure 3 shows a decision-making scheme that we chose in order to carry out our first experiments. We configured the wargame existing AI (the bootstrap AI) to control, on one hand its own camp, and on the other hand, corps commanders and their subordinate units for our camp. As a result, our system (the learning AI) controls only the army commander (the highest level commander) of our camp. When requested, the learning AI determines which order the army commander issues to each of his subordinates.

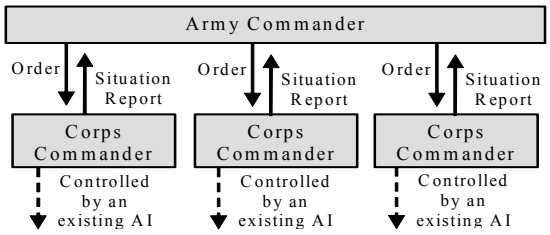

Figure 3: Decision-making scheme for our first experiments

At the beginning of each turn, in order to take a decision, the army commander needs to know the situation of his corps units and some information about the enemy units. To this end, he uses an abstracted state representation that we designed based on a semi-automated terrain analysis of the game map and on a specific scenario configuration for this map. As a result, the game map was divided into six zones based on tactical analysis of terrain features, the scenario objectives and the initial situation of the armies.

In terms of RL, we initially use a table as memory for the learning game strategy. This type of memory is simple to implement, but is enormously costly in terms of memory space. Consequently, the chosen representation does not contain much detailed information. The state representation is composed of two main groups of data: (1) a corps description (i.e.: its location in one of the 6 zones above), its strength and fatigue levels, and quality rating, (2) its environment (the total strength of friendly and enemy units in each map zone). One interesting feature of our representation is that two different units can share the same state if they have the same abstracted description and environment.

The possible actions for a commander are based on two attributes: an order type (extreme attack, attack, no order, defend and extreme defend) and a target location on the map (hexagon). In order to constraint the action space, we chose a very small number of key hexagons (usually one) for each one of the six zones.

The RL algorithm used by our system is SARSA(λ) with replacing eligibility traces and clearing traces for untaken actions (Sutton and Barto 1998). It is applied to learn state-action values $Q(s,a)$ by repeated application of an incremental update rule:

$$Q_{t+1}(s,a) \leftarrow Q_t(s,a) + \alpha\delta_t e(s,a), \text{ for all } s,a$$

where $\delta_t = r_{t+1} + Q_t(s_{t+1},a_{t+1}) - Q_t(s_t,a_t)$ is the temporal difference error and $e_t(s,a)$ is the eligibility trace for the state-action (s,a).

The reward function r is computed by our system as the change in the game score between the current and next game turns.

First results

The first results of our experiments indicate that our system has made good progress for the first ten thousand episodes (i.e. completed learning games) with an important improvement in average score (see Figure 4). After the first ten thousand episodes, our system does not improve its score anymore.

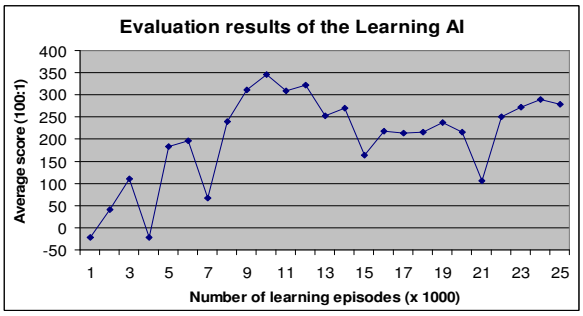

Figure 4: Progress evaluation of the Learning AI in a total of 25000 episodes (100 evaluations for each 1000 episodes)

In order to compare our learning AI with other kind of agents (a random and the non-adaptive bootstrap AI), we evaluated these two agents by playing against the non-adaptive bootstrap AI for some thousands of games. While the random agent obtains an average final score of -14 points, our learning agent so far reaches an average score of 345 points (average of 100 evaluations after 10000 episodes of learning), which places it close to the performance of the bootstrap AI, estimated at 350 points on average. These scores are related to the victory values of the game scenario, and these values range from major defeat to major victory. The value description and an agent comparative performance graph are showed in the Figure 5.

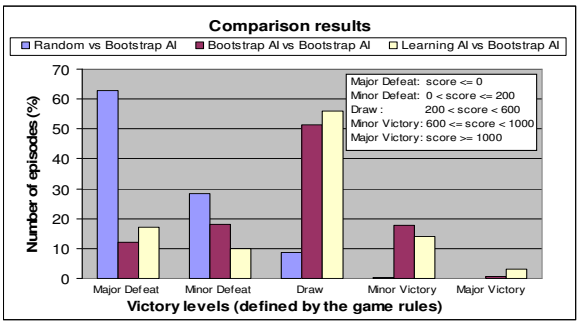

Figure 5: Histogram of victory levels for 3 types of AI against the bootstrap AI

We can see that the learning AI has managed to approach very closely the level of performance of the bootstrap AI. It is a very significant result to achieve the performance of an existing rule-based system with a learning system despite the limitation resulting from our choice of a very simplified representation of game situations.

Moreover, we can detect that our system has learned some basic strategy: early in the learning experiment, its decisions were completely inconsistent (indeed almost random), while now, after thousands of episodes, the learning AI consistently orders its units towards some strategic map points that bring the friendly units to the scenario objectives. We can observe too that the friendly units take much more risk in order to capture the most important objectives, while in the first ten thousand episodes they took more precaution. Although they still fail dramatically at some occasions now, on average they achieve higher scores. This indicates that long-term benefits of risky actions have been learned and are reflected in the adopted strategy.

We have short-term plans of an experimental nature that are expected to lead further improvements. However, reaching satisfying performances, and expanding the scope of learning to larger and larger parts of the decision-making process will require some more fundamental upgrades to our approach. Some of them are outlined in the next section.

Conclusions and Future Works

We proposed a new approach to the semi-automatic generation of a game AI using Reinforcement Learning techniques. To deal with the large scale and high complexity of the underlying game, our approach uses a bootstrapping of the learning process which allows to progressively learn a challenging game strategy in stages, tackling a smaller learning sub-problem at a time. The decomposition into sub-problems is based on the underlying hierarchical structure of the game as well as on some information abstraction appropriately designed for each level of the control hierarchy.

We are currently applying our bootstrapping methodology to the battleground series of wargames. In this paper, we reported the results of our first experiments, where the goal was to learn a high-level strategy for the army commander based on a tractable abstracted representation of the state of the game. The results so far are quite encouraging, and we are confident that with a more detailed representation and a capacity for approximation, further improvement can be achieved in the quality of the learnt strategies.

In future work, we plan to apply our methodology for learning strategies of low level commanders. In this case, it will become crucial to coordinate the decision making process of various units. This is an interesting challenge, and an active research topic in the field of Reinforcement Learning (Guestrin, Lagoudakis and Parr 2002).

Moreover, it is not feasible to control all levels of the hierarchy based on a tabular representation for the learning memory (because of the number of states to consider), we therefore have to tackle further the issue of generalization of the state representation. This can be done in two directions: abstracting the state representation based on domain knowledge (such as military terrain analysis), or via an automated approximation mechanism. We are currently exploring these two directions in parallel.

Acknowledgements

Charles Madeira's PhD work is funded by a scholarship from CAPES, Brazil within the framework of SMART-E's, a collaborative project between UFPE, Brazil, and UPMC, France, sponsored by CAPES-COFECUB.

References

Corruble, V., Madeira, C., and Ramalho, G. 2002. Steps Toward Building a Good AI For Complex Wargame-Type Simulation Games. In Proceedings of The 3rd International Conference on Intelligent Games and Simulation, London, United Kingdom.

Corruble, V. 2000. AI approaches to developing strategies for wargame type simulations. AAAI Fall Symposium on Simulating Human Agents. Cape Cod, USA.

Guestrin, C., Lagoudakis, M., and Parr, R. 2002. Coordinated Reinforcement Learning, In Proceedings of The Nineteenth International Conference on Machine Learning, 227-234, Sydney, Australia.

Mitchell, T. 1997. Machine Learning. Sing.: McGraw-Hill.

Nareyek, A. 2004. AI in Computer Games. ACM Queue. 1(10).

Rabin, S. eds. 2002. AI Game Programming Wisdom. Charles River Media.

Sutton, R. S., and Barto, A. G. eds. 1998. Reinforcement Learning, An Introduction. MIT Press.

Whiteson, S., Stone, P. 2003. Concurrent Layered Learning. In The Proceedings of the second international joint conference on Autonomous agents and multiagent systems. 193-200. New York, NY, USA: ACM Press

Zucker, J.-D., Bredèche, N., and Saitta, L. 2002. Abstracting Visual Percepts to learn Concepts. Symposium on Abstraction Reformulation and Approximation. SARA'2002. Canada.