PhD Thesis Proposal

Active Behaviour Learning: Mapping Play Strategies into Action Sequences

Student: Fahong Li

Thesis committee: Dr. Robert J. Woodham (supervisor)

To be assigned.

To be assigned.

Department of Computer Science

The University of British Columbia

February 8, 2007

Abstract

its definition

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Chapter 2

Related work

2.1 Learning in RCS2DS

[1] presents a flexible team member agent architecture and a general-purpose layered machine learning paradigm which allows learning at each level of a hierarchy composed of complex tasks. The RoboCup Soccer simulator environment is used as the major testbed. The author identifies the characteristics of the environment and define it as a PTS (Periodic Team Synchronization) domain. Roles, formations and set-plays are used to compose teamwork structure, and paradigms of communication among agents with a single unreliable low-bandwidth channel are also analyzed. In the layered learning paradigm: neural network is used to learn an individual soccer skill, ball interception; decision tree is employed to learn a multi-agent behavior, pass evaluation; and a new algorithm TPOT-RL (Team-Partitioned, Opaque-Transition Reinforcement Learning) is taken to learn a team behavior, pass selection. The algorithm TPOT-RL is also tested in the domain of network routing. A fully functioning multi-agent system embodying the algorithms and architecure was implemented and participated in the RoboCup Soccer competitions. The author also suggests to use observational reinforcement learning for strategic positioning and memory-based algorithm for strategic adaptation. As the author stated one problem with the layered learning is the error propagation or credit assignment among layers.

[2] defines the coaching problem, i.e., an automated coach agent providing advice to one or more automated advice-receiving agents, and explores solutions to the challenges posed in the prolem, such as learning and us-

ing models of the environment, adapting advice to receivers' peculiarities, representing advices and modeling opponents. The author co-developed the standard advice language CLang for the simulated robot soccer environment, and proposed a multi-agent plan representation MASTN (Multi-Agent Simple Temporal Network) and an associated distributed plan execution algorithm. However, in the current implementation of the algorithm, the agents do not take over actions which were originally assigned to others but not executed successfully by them. Based on two assumptions: 1) the variation in opponents can be approximately expressed in a reasonably sized set of models from which to choose; and 2) the output of an opponent model does not explicitly depend on the positions of the agent's teammates, the author built several models of opponent movements in setplays while game stops.

[3] introduces a constraint-based agent architecture to address the issues of an agent's: synchronization with and response to the dynamic environment, and operating proactively and appropriately in the environment. Intentions represented in the form of constriants are attributed to the agent to achieve its resource-bounded deliberation. The architecture adaptively schedules deliberation processes so that the agent can evolve its action with the natural frequency of the environment's dynamics. The architecture is modular and its execution is interruptible to produce quality-varying solutions. Transcribed in a dynamic and probabilistic multi-agent environment, a theory of intentionality is presented to abstract behaviors and to facilitate agents' behavior generation and recognition. The architecture is implemented and tested as agents playing in the RoboCup Soccer Simulation League: the agents' internal representations of the world are compared with the true states of the environment and the agents' successes in dynamic situations with varying resource requirements and uncertain percepts are evaluated. The internal representation model has yet been integrated into the whole decision process of the agents.

[4] presents a formalization of team strategy and concepts of Situation Based Strategic Positioning (SBSP) and Dynamic Positioning and Role Exchange (DPRE) for homogeneous agents to collaborate against opponents in dynamic, real-time and uncertain environments. The formalized team strategy is composed of a set of tactics and several possible agent types. A tactic consists of formations that are applied in different situations. A formation assigns each agent its agent types and its positions in the field. The strategic positioning of one agent depends on the situation and the positioning of other agents assigned in the formation. Agents take more reactive behaviours, i.e.,

domain-specific high-level and low-level skills, when they are in active situations, which are identified out of strategic situtations. Agents are also able to switch their positions and roles (specific behaviours) at run-time, within one formation. A high-level decision module is used to decide an agent's current tactic, formation, role and action at a given moment. The FC Portugal RoboCup Soccer team implemented an agent architecture embodying this team strategy and concepts of SBSP and DPRE. It won all the opponent teams in both the European and the World RoboCup 2000, without losing any goal in any of the competitions.

2.2 Agent control architecture

[5] presents a soft real-time agent control architecture to generate schedules satisficing temporal, structural and resource constraints, to merge new goals with existing ones, and to detect and handle unexpected results from activities. The architecture uses TAEMS, a description language for hierarchical task decomposition, to model goals, subgoals and low-level activities with quantitative and probabilistic parameters. Given a TAEMS representation of a task, a planner DTC (Design-To-Criteria) and a scheduler POS (Partial Ordered Scheduler) evaluate current runtime context and execution characteristics to generate and rank a range of candidate plans and schedules, and select one from them. An execution subsystem executes activities in this plan, tracks the performance, and reschedules or resolves conflicts when appropriate. Though the authors claim the TAEMS structure and parameters for a task can be learned, it is difficult to come up with a reasonable decomposition of a complex task as required in TAEMS. In addition, when a failure occurs during the execution of some activity, the architecture can not indicate which factor(s) caused that failure.

[6] presents IDEA (Intelligent Distributed Execution Architecture), a framework to unify planning and execution with four major components: the domain model, the plan database, the plan runner, and the reactive planner. Deliberative planning is achieved through modeling and problem solving on the plan database within the framework. IDEA allows the specification of agents that operate within a guaranteed reaction time and supports flexible specification of reactive vs. deliberative agent behavior. It also defines a simple communication protocol among several agents which implement IDEA. In IDEA a layered system can be implemented as separate agents, one per

layer, each representing its interactions with the world in a model. At all levels, the model representation primitives and their semantics is the same. IDEA is used to supersede the functionalities of the Deep Space 1 Remote Agent (DS1 RA) [7]. The DS1 RA architecture includes three layers of functionality: a constraint-based planner/scheduler (PS), a reactive executive (EXEC), and a Model Identification and Recovery system (MIR) consisting of a model-based truth maintenance system with diagnosis and recovery module. These layers use different modeling languages and different ways specifying problem-solving control. Although this heterogenous approach has the advantage of tuning each module to maximize performance at that level, it is difficult to validate all the models and procedures and to insure no conflict among them.

[8] describes a flexible and reusable agent architecture STEAM to address teamwork requirements such as coordination and communication in complex, dynamic multi-agent domains. Based on the Joint Intentions theory [9], team members in the STEAM framework synchronously builds up a hierarchy of shared intentions, individual intentions and beliefs about others' intentions. They use decision theory to decide whether to communicate to get mutal belief while building or disbanding joint intentions. With explicit representation of team goals and plans, STEAM monitors team performance during the execution of the hierarchy of reactive plans, and reorganizes the team when an individual fails in fullfilling its responsibilities or no team member is covering a new task. The implementation supplements rules for teamwork to the SOAR agent architecture [10].

In order to generate intelligent agent behavior in complex, real-time and dynamic computer game worlds, [11] proposes an anytime planner A-UMCP (Anytime Universal Method Composition Planner) and an agent architecture, which supports the planner and deals with the game domain features such as real-time interactivity and complexity. The planner trades off planning time against plan quality by using hierarchical task networks and allowing its planning process interruptible at any time, even without the requirement of finding an initial solution first. It generates information to guide agent behavior rather than exact solutions to problems. The agent itself is responsible for translating the information into its executable primitives. This additional translation step between planning and acting is one of the weaknesses of the architecture. The weaknesses also include a lack of inter-agent communication features and a lack of proactive intelligence.

2.3 Active learning

Given the freedom of choosing future training data based on those that have been used, [12] proposes a general approach for active learning to minimize the number of training data required and thus to reduce the cost of gathering them. This approach first defines a model and an associated model quality or loss function appropriate for the learning task at hand. Then it chooses a method to compute the potential model loss given a potential query, and finally asks the query which causes the lowest potential model loss. It has been applied in three areas of machine learning: classification with Support Vector Machines, parameter estimation and causal structure discovery in static Bayesian Networks. Empirical results show the active learning technique can significantly reduce the need for training data. There are still many aspects for the active learning method to explore, such as dealing with situations involving missing data values, hidden variables, high-dimensional problems and temporal domains.

[13] presents a framework and objective functions for active learning to close the data-gathering loop in three fundamental Hidden Markov Model (HMM) problems: learning individual states, learning the most likely path of hidden states given some observations, and learning the model which generated the data. The active learning HMM is equivalent to a standard HMM except some observations in the former are hidden and queryable. In the framework, the loss function is determined by what one dislikes about one's current belief state. Uncertainty about hidden variables and expected error on tasks (e.g., mislabelling A into B) are two common types of them. The Value of Information (VOI) is defined as the expected reduction in loss (uncertainty or cost) once the information is observed or provided as an answer to some query. The primary bottleneck of the framework is to incorporate the information learned from the quries into the HMM's beliefs, since the VOI computations are fast enough.

Assuming the model class used by an active learner can actually describe the true distribution underlying the data and the model has no hidden, continuous variables, [14] generalized the Query by Committee approach to unsupervised active learning tailored for domains with large number of random variables. As the standard information gain (Jensen-Shannon divergence) requires a committee size growing exponentially with the number of independent subdomains involving uncertainty, the authors propose a new additivite divergence measure to quantify the degree of disagreement in terms of average Kullback-Leibler divergence between all pairs of committee members, so that the lowest committee size required is given by the subdomain with the largest uncertainty. A bootstrap approach for committee selection is also presented.

2.4 Game (Opponent) AI

[15] proposes a taxonomy of domain knowledge in simulated soccer to facilitate the definition of similarity measures between two situations. The taxonomy classifies domain knowledge at the first level into two categories: distributional knowledge and virtual or derived attributes. Distributional knowledge is about variables' range and distribution or ordering of nominal feature values. Virtual attributes have two types: matching knowledge, which can be taxonomies or continuous/discrete transformational knowledge, and inferential knowledge, which includes relations and contextual knowldege such as weights. A non-extended similarity measure only incorporates weighted Eculidean distances between players' and balls' position and velocity vectors. An extended similarity measure takes various knowledge-rich virtual attributes and their weights (which can be learned) into account. With respect to the extended similarity measure, players in two situations are first matched, and if the two situations are similar enough, the high-level action (such as shoot-on-goal) taken by player A in one situation will be given as the predcited action of player A's counterpart in the other situation. The difficulty of this similarity-based opponent modelling lies in defining the similarity measure which incorporates reasonable virtual attributes and appropriate weights for different situations.

A case-based reasoning system CAT (CAse-based Tactician) is designed in [16] to win against randomly selected opponents in WARGUS, a real-time strategy game using the open-source engine STRATAGUS [17, 18]. CAT uses three sources of WARGUS domain knowledge, i.e., a state lattice, a set of tactics (subplans) for each state, and cases that map game situations to tactics and their performance, to facilitate case acquisition and tactic selection. It is tested in TIELT [19] and the results show that CAT learns to play significantly better than the best performing genetically evolved counter-strategy against WARGUS opponents.

[20] Online Adaptation of Game Opponent AI in Simulation and in Practice

2.5 Constraints and AI planning

[21] Constraints and AI Planning

Chapter 3

Proposed solution

Chapter 4

Discussion and conclusion

Bibliography

[1] Peter Stone. *Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer*. MIT Press, 2000.

[2] Patrick Riley. *Coaching: Learning and Using Environment and Agent Models for Advice*. PhD thesis, Computer Science Dept., Carnegie Mellon University, 2005. CMU-CS-05-100.

[3] Jeffrey Montgomery. Situated observation and participation in multiple-agent systems. Master's thesis, Computer Science Dept., The University of British Columbia, November 2003.

[4] Luís Paulo Reis, Nuno Lau, and Eugénio C. Oliveira. Situation based strategic positioning for coordinating a team of homogeneous agents. In Markus Hannebauer, Jan Wendler, and Enrico Pagello, editors, *Balancing Reactivity and Social Deliberation in Multi-Agent System – From RoboCup to Real-World Applications*, number 2103 in Springer's Lecture Notes in Artificial Intelligence, pages 175–197. Springer, Berlin, 2001.

[5] Bryan Horling, Victor Lesser, Régis Vincent, and Thomas Wagner. The soft real-time agent control architecture. *Autonomous Agents and Multi-Agent Systems*, 12(1):35 – 91, Jan 2006.

[6] Nicola Muscettola, Gregory A. Dorais, Chuck Fry, Richard Levinson, and Christian Plaunt. Idea: Planning at the core of autonomous reactive agents. *Artificial Intelligence*, 2002.

[7] Nicola Muscettola, P. Pandurang Nayak, Barney Pell, and Brian Williams. Remote agent: To boldly go where no ai system has gone before. *Artificial Intelligence*, 103((1-2)):5–48, August 1998.

[8] Milind Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.

[9] H. J. Levesque, P. R. Cohen, and J. Nunes. On acting together. In *Proceedings of the National Conference on Artificial Intelligence*, pages 94–99, Menlo Park, Calif., 1990. AAAI press.

[10] URL. Soar. retrieved from http://sitemaker.umich.edu/soar on July 25, 2006.

[11] N. A. Hawes. *Anytime Deliberation For Computer Game Agents*. PhD thesis, School of Computer Science, The University of Birmingham, Birmingham, B15 2TT, November 2003.

[12] Simon Tong. *Active Learning: Theory and Applications*. PhD thesis, Stanford University, August 2001.

[13] Brigham Anderson and Andrew Moore. Active learning for hidden markov models: Objective functions and algorithms. In Luc De Raedt and Stefan Wrobel, editors, *Proceedings of the 22nd International Machine Learning Conference*. ACM Press, 2005.

[14] H. Steck and T. Jaakkola. Unsupervised active learning in large domains. In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI-02)*, pages 469–476, San Francisco, CA, 2002. Morgan Kaufmann Publishers.

[15] Timo Steffens. Similarity-based opponent modelling using imperfect domain theories. In Graham Kendall and Simon Lucas, editors, *IEEE 2005 Symposium on Computational Intelligence and Games (CIG'05)*, pages 285–291, 2005.

[16] D. W. Aha, M. Molineaux, and M. Ponsen. Learning to win: case-based plan selection in a real-time strategy game. In *To appear in Proceedings of the Sixth International Conference on Case-Based Reasoning*, pages 15–20. Chicago, IL: Springer, 2005.

[17] M.J.V. Ponsen, S. Lee-Urban, H. Munoz-Avila, D.W. Aha, and M. Molineaux. Stratagus: An open-source game engine for research in real-time strategy games, 2005. Washington, DC: Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence.

[18] URL. Stratagus: A real time strategy engine. retrieved from http://stratagus.sourceforge.net/index.shtml on August 3, 2006.

[19] URL. Tielt: Testbed for integrating and evaluating learning techniques. retrieved from http://www.tielt.org/ on August 3, 2006.

[20] Pieter Spronck, Ida Sprinkhuizen-Kuyper, and Eric Postma. Online adaptation of game opponent ai in simulation and in practice. In Quasim Mehdi and Norman Gough, editors, *Proceedings of the 4th International Conference on Intelligent Games and Simulation (GAME-ON 2003), ISBN: 90-77381-05-8*, pages 93–100, EUROSIS, Belgium, 2003.

[21] A. Nareyek, R. Fourer, E. C. Freuder, E. Giunchiglia, R. P. Goldman, H. Kautz, J. Rintanen, and A. Tate. Constraints and ai planning. *IEEE Intelligent Systems*, 20(2):62–72, 2005.