Adaptive coordination among fuzzy reinforcement learning agents performing distributed dynamic load balancing

David Vengerov, Hamid R. Berenji
vengerov@stanford.edu
berenji@ptolemy.arc.nasa.gov
Intelligent Inference Systems Corp.
Computational Sciences Division, MS: 269-2
NASA Ames Research Center
Mountain View, CA 94035

Alex Vengerov
abvenger@ramapo.edu
School of Administration and Business
Ramapo College of New Jersey
505 Ramapo Valley Rd.
Mahwah, NJ 07430

Abstract—

In this paper we present an adaptive multi-agent coordination algorithm applied to the problem of distributed dynamic load balancing. As a specific example, we consider the problem of dynamic web caching in the Internet. In our general formulation of this problem, each agent represents a mirrored piece of content that tries to move itself closer to areas of the network with a high demand for this item. Each agent in our model uses a fuzzy rulebase for choosing the optimal direction of motion and adjusts the parameters of this rulebase using reinforcement learning. The resulting architecture for multi-agent coordination among fuzzy reinforcement learning agents (MAC-FRL) allows the team of agents to adaptively redistribute its members in the environment to match the changing pattern of demand. We simulate the performance of MAC-FRL and show that it significantly improves performance over non-coordinating agents.

I. INTRODUCTION

The subject of multi-agent systems is characterized by a shift of attention from modeling individual agents to modeling interactions between agents and analyzing the effects of these interactions on performance of society as a whole. An essential form of interaction in multi-agent systems is that of distribution of tasks or resources among the individuals.

As summarized in [1], the main traditional approaches to task distribution are: 1. *Imposed allocation*: the superior agent tells other agents which tasks to perform. 2. *Allocation by trader*: special agents - traders or brokers - gather requests and bids for service and match them together for execution. 3. *Allocation by acquaintances*: assumes an acquaintance network where agents are aware of the capabilities of their neighbors. The requests for service propagate through this network until a match has been found. 4. *Allocation by contract net*: an auction-like protocol is used to communicate between the clients and the servers. At first, clients openly post descriptions of tasks to be performed. On the basis of these descriptions each server draws up a proposal describing the service it can render and its price. Each client receives the proposals, evaluates them, and awards the contract to the best bidder.

Allocation methods described above assume a predefined communication structure, which all agents have to follow. This approach works well in simple, structurally stable environments. In contrast, the emergent allocation methods use a principally different communication method that is signal-based rather than message-based. Signals do not have semantics and can be interpreted differently by different receivers depending on their context. In the emergent allocation method, agents learn the value of these signals in the context of their local environments. Emergent allocation and communication methods are beginning to receive more and more attention now, as researchers are turning to modeling complex systems embedded in uncertain and nonstationary environments.

The architecture for multi-agent coordination among fuzzy reinforcement learning agents (MAC-FRL) presented in this paper is an example of emergent allocation methods. In this architecture, agents combine the local information about their environment with the global information about the multi-agent team and learn the context-dependent value of this combination. The dynamics of a two-level architecture that uses global information to speed up individual learning was formally analyzed in [13]. The goal of MAC-FRL is to allow the multi-agent team continually redistribute its members in the environment in proportion to the instantaneous demand for ser-

0-7803-7280-8/02/$10.00 ©2002 IEEE

vice present in each area of the environment. As a testing example for MAC-FRL, we use the problem of distributed dynamic web caching in the Internet. This problem is of a great importance for the future of the Internet, with companies such as Akamai [2] building large commercial infrastructures for intelligent content redistribution. We present a simulation study and analysis of the issues that arise when content agents are trying to achieve the conflicting objectives of moving toward the highest demand area and achieving a good joint coverage of the overall network.

In section II we give a brief overview of reinforcement learning with function approximation. Section III presents the structure of a fuzzy rulebase agent used in MAC-FRL. Section IV presents equations of the fuzzy reinforcement learning algorithm used by individual agents. Section V describes the principal features in the domain of dynamic web caching that are relevant for our multi-agent coordination algorithm. Section VI describes the simulator of the dynamic web caching problem and describes implementation of MAC-FRL in this domain. Section VII describes our experimental results and section VIII presents conclusions and implications of our work. Section IX describes the related work on multi-agent coordination.

II. FUZZY REINFORCEMENT LEARNING

Reinforcement learning is learning the mapping from situations to actions so as to maximize a numerical reward signal. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover by trial-and-error the actions yielding the largest reward [3].

In many tasks to which we would like to apply reinforcement learning, most states will be encountered only once, which makes it very difficult to learn the value of these states. This will almost always be the case when the state space is very large or continuous. The only way to learn in these domains is to generalize the learning experience from previously encountered states to the similar ones that have not been visited yet.

Fortunately, generalization from examples has already been extensively studied, and totally new methods for use in reinforcement learning do not need to be developed. Instead, the generalization methods only need to be properly combined with reinforcement learning algorithms. The kind of generalization required is often called function approximation because it samples the desired function (e.g., a value function) over the previously visited states and attempts to generalize its value to all states.

In this paper we use local function approximation architectures based on feature extraction using fuzzy logic. However, the idea of MAC-FRL does not depend on the details of the individual learning algorithms and can work just as well with global function approximation methods such as neural networks.

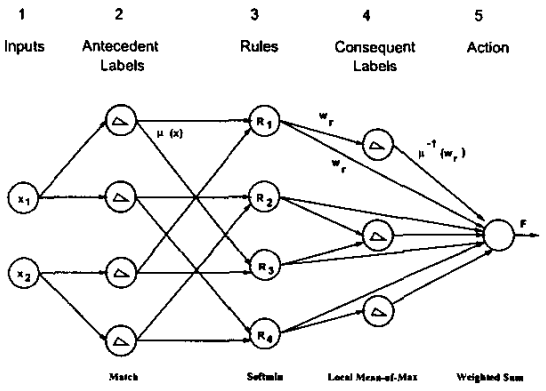

Figure 1: Structure of an FRL agent

III. AGENT STRUCTURE

In this section we show how Fuzzy Reinforcement Learning (FRL) can be used for developing heterogeneous rulebased intelligent agents with different expertise. For example, a knowledge-rich agent may include a large number of rules allowing it to deal with a great variety of situations, while a more operational or action-rich agent may have fewer rules but be more specialized in executing some specific tasks. Alternatively, agents can have the same number of rules but have them cover different situations that can arise, providing expertise in different aspects of autonomous missions.

The rulebase used by a FRL agent can be represented as a network with 5 layers of nodes, each layer performing one stage of the fuzzy inference process (see Figure 1). The connections are feedforward, with each node performing a local computation. The computational details of processing performed at each layer are given in [4].

The fuzzy rulebase approach to agent design has several unique advantages:

1. General human knowledge of the proper agent reactions in various situations can be easily imparted to the agents. This knowledge is naturally formulated in terms of rules, which can be used as a starting point for further refinement during agent learning.

2. Final rules at the end of a learning period can be easily understood and corrected by human experts.

3. By combining the so-called "actor-critic" reinforcement learning algorithm with fuzzy rulebases, the learning process of each agent is guaranteed to converge to optimality. A proof of this FRL convergence result is given in [5].

IV. LEARNING ALGORITHM

In our simulations we have combined the Q-learning algorithm of Watkins [6] with fuzzy logic function approximation. The Q-values are generalized across states by using a function approximation architecture $Q(x, a, r)$ for approximating $Q(x, a)$, where r is the set of all learned parameters arranged in a single vector. The basic parameter updating rule used by discounted Q-learning or Monte Carlo learning for such an architecture is presented in [7]:

$$r_t \leftarrow r_t + \alpha \delta_t \nabla_{r_t} Q(x_t, a, r_t), \tag{1}$$

where α is the learning rate and δ_t is the Bellman error used in the corresponding learning rule for the look-up table case:

$$Q(x_t, a) \leftarrow Q(x_t, a) + \alpha \delta_t. \tag{2}$$

For example, in the look-up table version of discounted Monte Carlo learning,

$$\delta_t = R_T + \sum_{\tau=t}^{T-1} \gamma^{\tau-t} g(\tau) - Q(x_t, a), \tag{3}$$

where $g(t)$ is the cost incurred at time t, γ is the discounting factor and the summation extends until the end of the episode. In the look-up table version of discounted Q-learning,

$$\delta_t = g(t) + \gamma \max_a Q(x_{t+1}, a) - Q(x_t, a). \tag{4}$$

In the general version of discounted Q(λ)-learning, equation (1) becomes:

$$r_t \leftarrow r_t + \delta_t \sum_{\tau=T_0}^{t} (\alpha\lambda)^{t-\tau} \nabla_{r_t} Q(x_t, a, r_t), \tag{5}$$

where T_0 is the time when the current episode began and δ_t is given by equations (3) or (4).

The analytical expression for approximating the Q-value using the fuzzy rulebase from Figure 1 is:

$$Q(x, a) = \sum_{k=0}^{K} q(k, a) \mu_k(x), \tag{6}$$

where $q(k, a)$ is the Q-value of taking the action a in the k-th fuzzy state s_k and $\mu_k(x)$ is the degree of membership of state x to s_k. If the action space is continuous, then equation (6) still applies after changing $\mu_k(x)$ to $\mu_k(x, a)$.

With $Q(x, a)$ given by equation (6), $\nabla_{r_t} Q(x_t, a, r_t)$ becomes $\mu_k(x_t)$. Thus, equations (1) and (5) can now be rewritten as matrix equations with each component given by:

$$q(k, a) \leftarrow q(k, a) + \alpha \delta_t \mu_k(x_t). \tag{7}$$

$$q(k, a) \leftarrow q(k, a) + \alpha \delta_t \sum_{\tau=T_0}^{t} (\alpha\lambda)^{t-\tau} \mu_k(x_t). \tag{8}$$

The above equations have a natural interpretation in the realm of fuzzy state aggregations: the Q-value of a fuzzy state-action pair (s_k, a) gets updated proportionally to its contribution to the Q-value of the state-action pair (x_t, a) in equation (6).

If the average cost formulation is used instead of the discounted cost formulation, then equations (7) and (8) still hold, except that δ_t in these equations is given for Monte Carlo learning by [3]:

$$\delta_t = \sum_{\tau=0}^{T} \gamma^{\tau-t} g(\tau) - Q(x_t, a) - \rho_t \tag{9}$$

and for Q(λ)-learning by

$$\delta_t = g(t) + \gamma \max_a Q(x_{t+1}, a) - Q(x_t, a) - \rho_t. \tag{10}$$

The quantity ρ represents the average reward per time step of the policy learned so far, to which the average reward from every state-action pair is compared. The quantity ρ is updated at every iteration according to

$$\rho_t \leftarrow \rho_t + \alpha \delta_t. \tag{11}$$

In the next section we describe the main characteristics in the domain of distributed dynamic web caching that are relevant for our study of multi-agent coordination in MAC-FRL.

V. DISTRIBUTED DYNAMIC WEB CACHING

With the exponential growth of hosts and traffic workload on the Internet, web caching has been recognized as the only viable solution for alleviating web server bottlenecks and reducing traffic over the Internet. Recently, there has been an increasing deployment of content distribution networks (CDNs) that offer hosting services to Web content providers. CDNs consist of servers distributed throughout the Internet that replicate provider content for better performance and availability than those in the older centralized approach. Existing work on CDNs has focused on techniques for efficiently redirecting user requests to appropriate CDN servers in order to reduce request latency and balance the load. However, little attention has been given so far to the more complex issue of dynamic redistribution strategies for Web content in order to match the changing pattern of user requests [8, 9].

In our simulation study, we use the following general formulation of the distributed dynamic content distribution in some CDN. The levels of demand for a certain information item at all locations in the CDN creates a demand

surface superimposed on the CDN. Agents that represent information content are moving to position themselves at the high points on this demand surface. They get rewarded at each time step based on the amount of demand they have satisfied. As more agents begin to service a certain area, the demand in that area slowly gets satisfied, and each agent begins to produce less and less benefit with time. We model this effect by having the demand surface slowly sink under each agent. If many agents stay for a long time in the same area, they will eventually satisfy all the demand there and will become useless, receiving little or no reward. Thus, there is a natural tradeoff present for agents in our model between satisfying the individual desire of moving to the highest demand area and satisfying the team goal of the proportional coverage of all areas in the CDN.

The next section describes the simulator we have developed for studying the general distributed dynamic load balancing problem. It also describes the implementation of MAC-FRL for this problem.

VI. DESCRIPTION OF THE SIMULATOR AND COORDINATION MECHANISM

We used a 2-D tileworld for simulating the most salient features in the dynamic content redistribution problem. Our tileworld consists of demand sources and agents in some locations. The number of demand sources in the world is kept constant. To ensure that agents learn a topology-independent coordination strategy, we allow each demand source to disappear at any time step with probability $1/MeanLife$ and reappear at a randomly chosen location. Newly appearing sources have a height (demand value) distributed uniformly between 0 and $MaxVal$. Each demand source j contributes the following amount to the demand potential of location i in the world:

$$P_{ij} = \frac{V_j}{1 + d_{ij}^2}, \tag{12}$$

where V_j is the value of the jth demand source and d_{ij} is the distance between the source and the considered location. The total potential of each location is computed as $P_j = \sum_i P_{ij}$. A graphical representation of this tile world is shown in Figure 2, with darker locations having a higher demand potential.

At every time step, the value of each source decreases by the amount of the total reward extracted by all agents from this source. An agent at location i extracts the reward from source j equal to P_{ij}. At every time step, agents choose which of the 8 neighboring locations they should move to. The goal of each agent is to maximize its average reward per time step. During the training process,

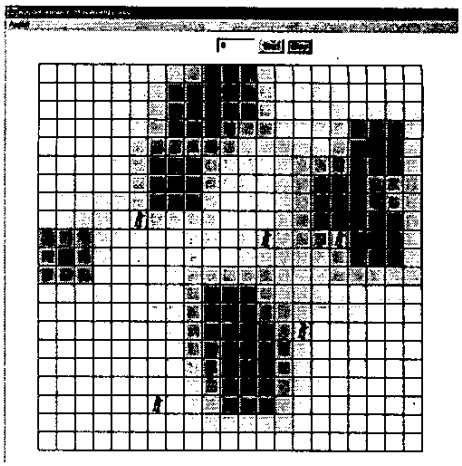

Figure 2: A potential surface model of the tile world

agents learn to evaluate locations in the world. The value of each location is the average reward per time step that can be obtained starting from that location and making the optimal choices thereafter.

When acting independently, agents evaluate each location based only on its demand potential. Then they move to one of the 8 surrounding locations with probability proportional to Q-values assigned to those locations. If an agents does not use any exploration and moves to the adjacent location with the highest potential, then the search procedure becomes equivalent to using the local gradient information for finding the highest point of the demand landscape. However, as our experiments show, using the gradient information can lead to individually optimal but socially suboptimal decisions.

In many domains, agents have only local visions of their environment. We simulate this by assigning a sensory radius to each agent. Only the demand sources within that radius can be used to compute the potentials of the surrounding locations.

In MAC-FRL agents have access to a single active blackboard. After submitting their locations to the blackboard, agents can request from it the value of the agent potential at any location in the tileworld. The agent potential is computed just like the demand potential, except that all agents are assigned the same value. The value of the agent potential at the considered location is used as an extra input variable for coordinating agents. Hence, in the spirit of emergent allocation methods described in the introduction, agents learn to assign appropriate context-dependent meaning to this variable.

The value of each input variable has been fuzzified into three labels: SMALL, MEDIUM, and LARGE. The

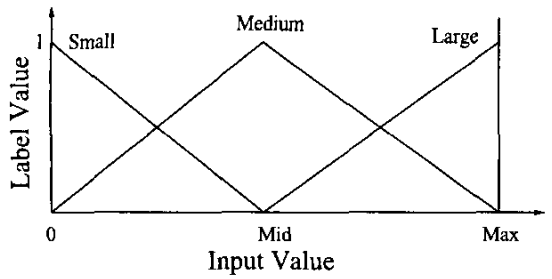

Figure 3: Fuzzy input labels used for each input variable.

shapes of these labels are shown in Figure 3. Hence, independent agents needed to learn the Q-values of three fuzzy rules while coordinating agents needed to learn the value of nine fuzzy rules. A sample fuzzy rule j for computing the Q-value of moving to location L_i by a coordinating agent is:

IF (demand potential at L_i is LARGE) AND (agent potential at L_i is SMALL) THEN (Q-value of moving to L_i is Q_j^i).

The final Q-value of moving to location L_i is given by equation (6): $Q^i = \sum_j \mu_j^i Q_j^i$, where μ_j^i is the degree to which fuzzy rule j applies for describing the location i. We compute μ_j^i using product inference by multiplying the label value of the demand potential by that of the agent potential. For example, assume that the maximum value for the potential surface is 100. Then, if the demand potential at location L_i is 75 and the agent potential is 10, then μ_j^i for rule j given above will be $0.5 \cdot 0.8 = 0.4$.

After moving to a new location, each agent computes the Q-value of the previous location using equations (7) and (10). The agent then distributes ΔQ among the conclusions of the fuzzy rules according to the Q-value each of them has contributed in equation (6).

VII. EXPERIMENTAL RESULTS

We ran experiments in a 20-by-20 tileworld with 10 demand sources and 5 agents. Agents have been trained for 1000 time steps and then tested for another 100 time steps. In these experiments, $MeanLife = 50$ and $MaxVal = 100$.

We experimented with two scenarios: unlimited sensory radius and a sensory radius equal to 5 units of distance. In both scenarios, all else being equal, coordinating agents learned to prefer locations with a smaller agent potential over those with a larger agent potential. That is, coordinating agents learned to avoid each other and ensure a good coverage of the environment. As a result, in both scenarios, agents combining Q-learning with our coordination mechanism obtained 50-100% better performance than agents using individual Q-learning or those choosing the direction of motion at random.

This is a significant result for our domain, since the domain is biased against agents using individual learning. This bias is best observed in simulations when agents used a limited sensory radius. In this case, independent agents obtained WORSE performance than agents choosing the direction of motion at random. In order to explain this phenomenon, we measured the average spread of agents in the tileworld. As expected, we found that the average spread of coordinating agents that learn to avoid each other is greater than that of random agents. However, we found that the average spread of independent agents with a limited sensory radius is much smaller than that of random agents. This can be explained by the fact that when independent agents happen to come near each other, all of them usually observe the same highest demand source hill, and consequently they all climb it, getting even closer together. As a result of this closeness, that area of the tileworld sinks quickly, and agents are left in a flat area with little reward per time step.

VIII. CONCLUSION

In this paper we described how to structure individual rule-based agents that can operate in continuous multi-dimensional state spaces. We then presented the MAC-FRL architecture for emergent coordination of actions among fuzzy reinforcement learning agents. MAC-FRL allows the multi-agent team continually redistribute its members in the environment in proportion to the instantaneous demand for service present in each area of the environment. A simulation study of MAC-FRL demonstrates its benefit in the challenging and practical domain of dynamic web caching, where teams of non-coordinating FRL agents obtain very poor results.

The MAC-FRL architecture has the following important features:

1. General human knowledge of the proper agent reactions in various situations can be easily imparted to the agents. This knowledge is naturally formulated in terms of rules, which can be used as a starting point for further refinement during agent learning.

2. Final rules at the end of a learning period can be easily understood and corrected by human experts.

3. It is independent of the reinforcement learning algorithm used by individual agents. For example, individual learning can be guaranteed to converge if the actor-critic reinforcement algorithm is used instead of Q-learning [5].

4. The extent of coordination in MAC-FRL is adaptive to the local state of the environment.

The last point implies that the MAC-FRL architecture utilizes the benefits of both centralized and decentralized approaches to decision making in multi-agent systems. In

the centralized approach, some central planning agency performs optimization and determines the actions each agent should take in order to maximize the team benefit. This approach is very efficient in simple problems, but fails to find even adequate solutions in more complex domains. Also, while satisfying the high-level strategic goals of the society, this approach ignores the low-level tactical goals of individual members. In the decentralized approach, each agent acts to maximize its own benefit, and the global behavior emerges from superposition of individual behaviors. This approach is more robust but is potentially very inefficient, since agents learn to accomplish their tactical goals while ignoring the strategic goals of the society. In the proposed emergent coordination architecture, agents receive the global information as a part of their state vector, and then learn the degree to which it should be utilized in the context of their local environments. Hence, this architecture allows a multi-agent system to adjust dynamically the balance between centralized and decentralized behavior, accomplishing both strategic and tactical goals. The simulation results and analysis conducted in our work demonstrates the effectiveness of this approach to emergent multi-agent coordination.

IX. RELATED WORK

To the best of our knowledge, the existing work on coordination of actions in multi-agent reinforcement learning does not allow agents to observe any information about each other (e.g. [10, 11, 12]). That is, agents learn to implicitly coordinate their actions by using the environmental reinforcement signal, which reflects the actions of other agents.

For example, Weiss [10] uses multi-agent reinforcement learning to solve a job assignment problem. In his formulation, different agents can have different execution times for the same job. However, Weiss uses a very simple coordination strategy, where a job that can be executed by several available agents is always assigned to the agent with the shortest execution time for this particular job.

Sen and Sekaran [11] have used reinforcement learning to coordinate actions between two agents in a simple resource sharing problem. However, they assumed that one agent has already applied some load distribution over a fixed time period, and the other agent is learning to distribute its load on the system without any knowledge of the current distribution. Therefore, the second agent is aware of the actions of the first one only through the reinforcement signal reflecting the state of the environment.

Arai and Sycara [12] have used multi-agent reinforcement learning for solving a pursuit game with multiple hunters and preys. However, no explicit coordination algorithm was used and agents had no knowledge about each other. As a result, the learned policy started converging in terms of its performance only after about 500,000 time steps, which is impractical in most real world problems.

REFERENCES

[1] Ferber, Jacques. *Multi-agent Systems: An Introduction to Distributed Artificial Intelligence.* Addison-Wesley, Harlow, England, 1999.

[2] Akamai. www.akamai.com

[3] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*, MIT Press, 1998.

[4] H. R. Berenji and P. Khedkar. "Learning and tuning fuzzy logic controllers through reinforcements", *IEEE Transactions on Neural Networks*, Vol 3:5, 724-740, 1992.

[5] Hamid R. Berenji, David Vengerov, "On convergence of fuzzy reinforcement learning," Proceedings of the 10th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2001.

[6] Christopher J. C. H. Watkins. *Learning from Delayed Rewards.* PhD thesis, King's College, Cambridge, UK, 1989.

[7] D. Bertsekas and J. Tsitsiklis. *Neuro-Dynamic Programming*, Athena Scientific, 2000.

[8] L. Qiu, V. Padmanabhan, and G. Voelker, "On the placement of Web server replicas," Proceedings of IEEE INFOCOM '01, Anchorage, Alaska, April 2001.

[9] G. Barish and K. Obraczka. "World wide web caching: Trends and techniques." *IEEE Communications Magazine*, Vol 38:5, pp.178-184, May 2000.

[10] G. Weiss. "A multiagent perspective of parallel and distributed machine learning." Proceedings of the 2nd International Conference on Autonomous Agents, pp. 226-230, 1998.

[11] Sandip Sen and Mahendra Sekaran. "Individual learning of coordination knowledge," *Journal of Experimental & Theoretical Artificial Intelligence*, Vol. 10, pp. 333-356, 1998.

[12] Sachiyo Arai, Katia Sycara and Terry R. Payne "Multi-agent Reinforcement Learning for Scheduling Multiple-Goals," Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS'2000).

[13] A. Vengerov. *Introduction to Harmonomics and Holistic Engineering: Application to Electronic Business and Systems Development.* Manuscript.