Proceedings of the
IEEE Intelligent Vehicles Symposium 2000
Dearborn (MI), USA • October 3-5, 2000

Multi Agent Collaboration Using Distributed Value Functions

Enrique D. Ferreira

Institute for Complex Engineered Systems
Carnegie Mellon University
1201 Hamburg Hall
Pittsburgh PA, 15213-3890
edf+@cmu.edu

Pradeep K. Khosla

Electrical and Computer Engineering Department
Carnegie Mellon University
1106 Hamerschlag Hall
Pittsburgh PA, 15213-3890
pkk@ece.cmu.edu

Abstract

In this paper we present the use of distributed value function techniques to reach collaboration in a multi-agent system. We apply this method in two different simulation environments: a mobile robot planning/searching task and an intelligent traffic system in an urban environment. In the case of the intelligent traffic system, results show an improvement with respect to a standard fix-time controller and local adaptive controllers. Trajectories for optimal search in an obstacle environment are obtained in the mobile robot case. Some variations to the actual algorithm are pointed out to suit our cases. We conclude discussing our future work.

Keywords Reinforcement learning, distributed system, mobile robot, traffic control.

1 Introduction

A significant part of real-world problems are difficult to solve because of their complex behavior and interactions. Other set of problems are difficult just because of their size. A logical approach is to partition a large problem into small ones that may be more manageable. However, that is not so simple to do when there is strong coupling between the subsystems. Adaptive and learning controllers may be able to model the interactions and generate useful responses. Reinforcement learning [8] (RL) is able to do just that by looking at the reaction of the environment to the actions the controller applies. Still, standard RL needs to work with the complete state space. In [6], a method to distribute reinforcement learning knowledge through different agents is introduced, called distributed value functions, and applied to a power grid system. In this paper we analyze some of the results of the algorithm introduced and present the application of the distributed value function idea to two different applications related to inter-vehicle communication and coordination and intelligent traffic systems. These are two very active areas at the moment. Large cities have been presenting traffic problems since quite some time [1]. A centralized approach to monitor and control is widely used but as systems becomes more complex and increase in size, scalability becomes an issue not easily solvable. Inter-Vehicle communication has been pushed forward due to its technical feasibility in the near future [4, 7] and its potential benefit, taking part of the burden out of large infrastructure investments, from military applications to the common everyday driver.

This paper is organized as follows. Section 2 explains the distributed value function approach and algorithms. In section 3 a multi-robot and a intelligent transport system example are developed in simulation to illustrate the algorithms. Finally, section 4 summarizes our results and points out future lines of research.

2 Distributed Value Function

We are considering problems that have a large state/action space, making a distributed control solution desirable or even necessary. If reinforcement learning is also appropriate, the option of using distributed value functions may be selected.

The standard reinforcement learning framework [8] considers a system interacting with its environment by applying actions from a (finite) set $\mathcal{A}$ and getting a reinforcement signal R back. The value function at state $x \in \mathcal{X}$ satisfies the Bellman equation:

$$V(x) = \max_{a \in \mathcal{A}} (R(x,a) + \gamma \sum_{x' \in \mathcal{X}} p(x'|x,a) V(x')) \qquad (1)$$

where $V(x)$ is the value function at state x, γ is a discount factor and $p(x'|x,a)$ is the state transition probability function. The solution to this equation corresponds to the optimal value function and is given by the expected value of the discounted sum of future rewards for the optimal policy

$\Pi^* : X \longrightarrow \mathcal{A}$ by

$$V^*(x) = E\left(\sum_{t=0}^{\infty} \gamma^t R(x_t, a_t)\right) \quad (2)$$

In the case of a distributed system, such us multi-robots environments, with weak coupling between the different agents in it, the state space could be partitioned making a distributed representation of the reinforcement algorithm more appropriate. The distributed system S can be represented in the following way. Let S be composed of N agents. Each agent C_i have access to a subset $\mathcal{X}_i$ of the state of S and can apply actions in a subset $\mathcal{A}_i$ receiving a local reward R_i. In order to coordinate the actions of the different agents, certain information is passed between neighboring agents. Let the Distributed Value Function $V_i(x)$ for agent C_i at state $x \in \mathcal{X}_i$ be defined by:

$$\begin{aligned} V_i(x) &= \max_{a \in A_i} (R_i(x,a) + \\ &\quad + \gamma \sum_j f_{ij} \sum_{x' \in X} p(x'|x,a) V_j(x')) \end{aligned} \quad (3)$$

where f_{ij} are weighting factors for the value function at neighbor agent C_j over agent C_i. Figure 1 illustrates the idea of distributed value functions. The recursive nature of the equation allows non neighbors to interact making the solution optimal in a non-local sense. For the optimal policy, it can be shown that (3) is equivalent to the algebraic system

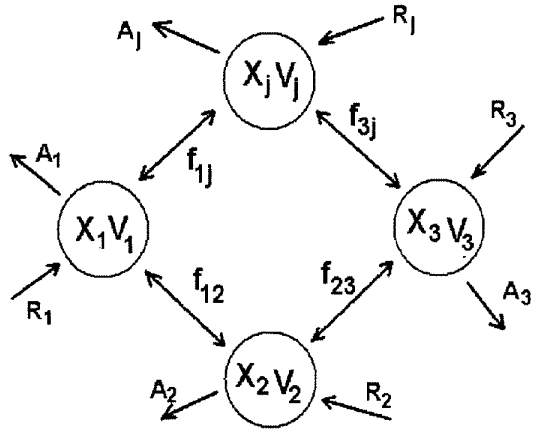

Figure 1: Idea behind distributed value functions.

$$(I - \gamma f_{ii} \Pi^*_{ii}) \mathbf{V}_i = \mathbf{R}^*_i + \gamma \sum_{j \neq i} f_{ij} \Pi^*_{ij} \mathbf{V}_j \quad (4)$$

for all i, where $\mathbf{V}_i$ is a vector stacking the values of $V_i(x_i)$ for all $x_i \in \mathcal{X}_i$, $\mathbf{R}^*_i$ stacks the local reinforcement signal for the optimal policy and Π^*_{ij} is the transition probability matrix for neighbor agent j given agent i state/action pair.

To handle the problem of having only access to local information Schneider et al. [6] proposed the use of Q-learning to compute $V_i(x)$. The algorithm first computes the Q function, dependent of the state/action pair,

$$\begin{aligned} Q_i(x_i, a_i) &\leftarrow (1-\alpha) Q_i(x_i, a_i) + \\ &+ \alpha (R_i(x_i, a_i) + \gamma \sum_j f_{ij} V_j(x'_i)) \end{aligned} \quad (5)$$

where α is the learning factor. The value function for agent C_i is given by

$$V_i(x_i) = \max_{a \in \mathcal{A}_i} Q_i(x_i, a) \quad (6)$$

For the optimal policy, after substituting (5) into (6), assuming $f_{ii} = f$ for all i and doing some manipulation, it can be shown that

$$V_i(x_i) = \sum_j \sum_{t=0}^{\infty} \gamma^t f^t(i,j,t) R^*(x_{j,t}, a^*_{j,t}) \quad (7)$$

is a solution for (6). It means that the distributed values maximize a weighted sum of future rewards over all agents. The weights depend on the coefficients f_{ij} selected before. Furthermore, if f_{ij} is symmetric, the sum of all V_i will be an equally weighted sum of discounted future distributed rewards over all the system. It remains to show the convergence of the Q-learning algorithm and the convergence of the value functions to (7). Also, the influence of the selection of the weight coefficients needs to be addressed to try to find the optimum set of coefficients to improve the solution.

3 Experiments

The distributed value function control algorithms have been implemented using a multi-agent software architecture developed by Carnegie Mellon University Advanced Mechatronics Laboratory (AML) group called *Port-Based Adaptable Agent Architecture* (PB3A) [3]. PB3A is a Java-based programming framework that aims to facilitate the development and deployment of self-adaptive, distributed, multi-agent applications. This distributed, multi-agent architecture allows systems to be created with the flexibility and modularity required for the rapid construction of software systems that analyze and dynamically modify themselves to improve performance. This architecture provides user-level access to the three forms of software adaptability: parametric fine tuning, algorithmic change, and code mobility. PB3A has been developed basically for mobile robot systems but its flexibility makes it easy to adapt to other multi-agent systems like intelligent traffic monitoring and control.

3.1 Urban Traffic Control

Using PB3A, the Q-learning approach to compute distributed value functions is implemented for a basic intelligent transportation system. In a urban traffic situation, adaptive traffic light systems are usually setup using global optimization algorithms. Here, we apply the distributed value function idea to compute the signal changes for each intersection controller. For each intersection, the local state consists of the signal group values, the number of vehicles in each lane coming to the intersection and the times since the last signal change for each signal group. The reward chosen is the time the vehicles are waiting for the green light. We should note here that, with this reward, we should use min instead of max in (6). Nevertheless, all the previous results are valid without any special changes. For comparison purposes different types of intersection controllers were designed: fixed-time controllers, periodically switching between red, yellow, and green lights; adaptive controllers based on local probabilities and queues; and the distributed value function controller using Q-learning techniques.

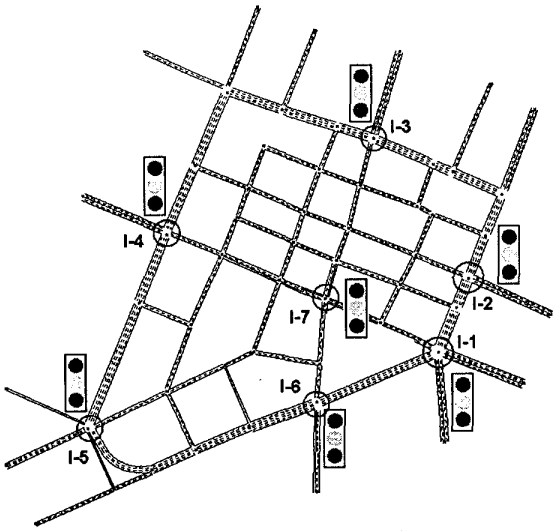

Figure 2: Road Model of Penn-Circle area.

The controllers are applied in simulation to a small section in the urban area of the city of Pittsburgh PA, called Penn Circle. The application came as a result of a collaboration in a Community Project to improve the traffic in the area. Penn-Circle is a four-lane, counterclockwise ring that constitutes one of the main passages for drivers coming from the eastern suburbs to get to downtown Pittsburgh.

The construction of Penn-Circle improved the traffic to downtown but caused heavy damage to the businesses in the area and completely changed the status of the neighborhood in a relative short period of time. Besides, due to the increase in traffic volume over the years, congestion problems are occurring in specific areas of the circle at peak hours. The congestion situation needs to be studied together with a major restructure to revive the area.

Our goal is to study the traffic patterns and current intersection-controller quality compared to more advanced controllers. With PB3A, it was possible to deploy the whole system very quickly and connect it with the traffic simulator ARTIST developed by Bosch [5]. Figure 2 shows the road model of the Penn-Circle area developed in ARTIST, with the locations of the main signal lights. Figure 3 shows a sketch of how the different blocks of the overall system are connected together. Seven intelligent intersection controllers are used. Each of them communicates with their topological neighbors. To simulate

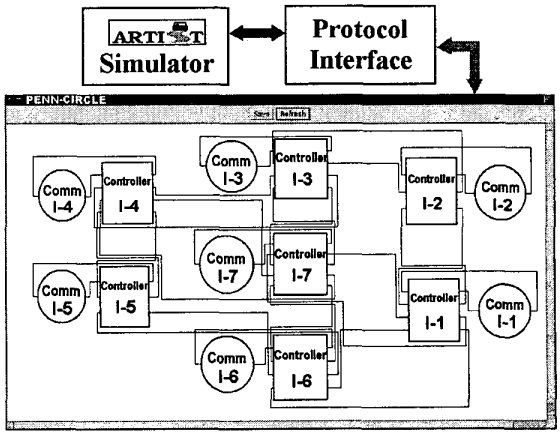

Figure 3: Simulation system diagram.

the traffic in the system, vehicle counts data provided by the City of Pittsburgh Dept. of City Planning and Trans Associates, Inc., are used to generate origin-destination matrices for morning and evening rush hour times. Vehicle paths are computed when each vehicle is initialized, varying between shortest path and shortest estimated time. The simulator may also recompute the path dynamically if some conflict arises because of congestion or incidents. Fix-time optimal signal changes were developed using the traffic data gathered. A local decentralized adaptive controller developed together with ARTIST [5] for the city of Oberhausen, Germany and adapted to Penn-Circle is also employed for comparison purposes.

Q-learning is applied until convergence of the distributed value functions. ARTIST is run on a Sun Sparc 20, dual processor while the PB3A Q-learning algorithm is run on an Intel 400 MHz dual pentium running RedHat Linux.

The process took about 10000 simulations of 15 minutes of traffic. The discount factor $\gamma = 0.95$ and the learning coefficient α is setup initially to 0.15 and decay over time. The weight factors f_{ij} are setup to compute the average of the value of each agent and its neighbors. After training, test is done with the learned policy and compared to the performance of the other controllers mentioned above. Figures 4 to 6 show the results. Both adaptive controllers outperform the fix-time controller. reducing the traffic volume in the system and improving the number of stops a vehicle makes in its way through the area. PB3A distributed controller does a better job than the adaptive controller in reducing the network vehicle density at the cost of the percentage of stopped vehicles.

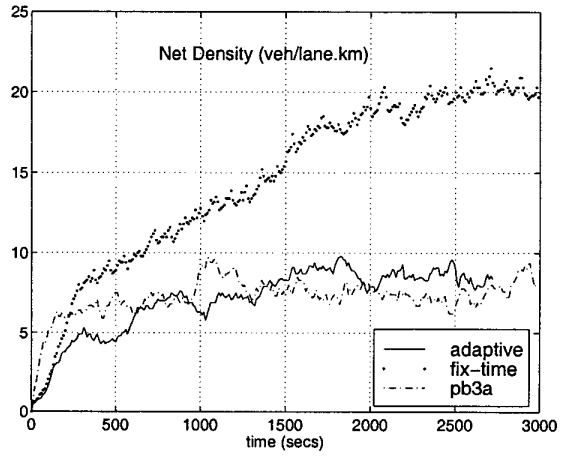

Figure 4: Traffic density over time.

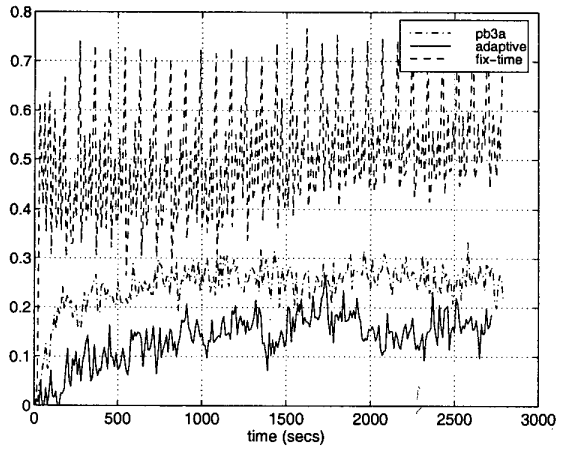

Figure 5: Percentage of non-moving cars over time.

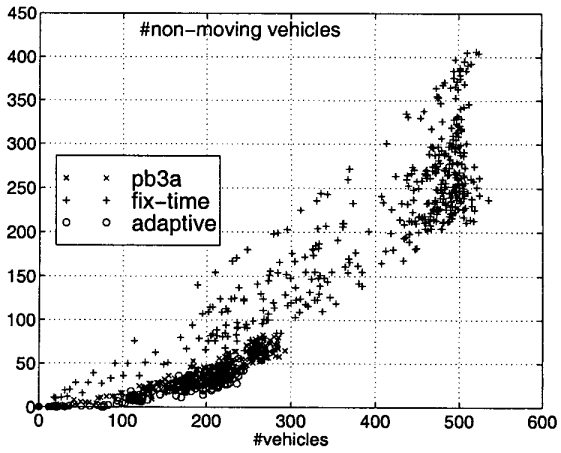

Figure 6: Vehicle distribution.

3.2 Multi-Vehicle Planning

In the case of the mobile-robot system, PB3A is interfaced with RAVE [2], an environment to combine real and virtual robots also developed at Carnegie Mellon University AML group.

To illustrate the distributed value function approach we selected a team search and rescue mission. Given an area, a team of mobile robots have to coordinate its movements to search for a party within the area in the minimum time possible.

Each robot knows its position, its sensor range and the means to recognize the searched object. It is also assumed that all the robots have a rough map of the environment in which they are performing the search mission. The actions are the possible movements of the robot (forward, backward, left and right turns). The reward signal is the new area covered by the robot in its movement. An additional negative reward would be issued for bumping into other objects.

However, this case is a little different from the standard algorithm shown in the previous section because the reward for agent i, the new area covered by the robot in its trajectory, depends on the relative position to its neighbors. It can be thought as a global reward, hence not fully distributed. To maintain the decentralization the communication between neighbors must include the value function plus their positions or range information to be able to compute the intersection of their sensor ranges. To keep the same framework we augment the local state by adding the neighbor states. There is still another detail to take care of. We must use a window of values to keep track of recently visited states. Further augmenting the state space also takes care of the mathematics although increasing the

size of the problem.

To train the team, for every trial, an object to be searched is randomly generated. The team is randomly setup on the location as well. Trials end when the searched object is localized. Each virtual robot is modeled to simulate a Pioneer II robot as shown in Figure 7. Different environment

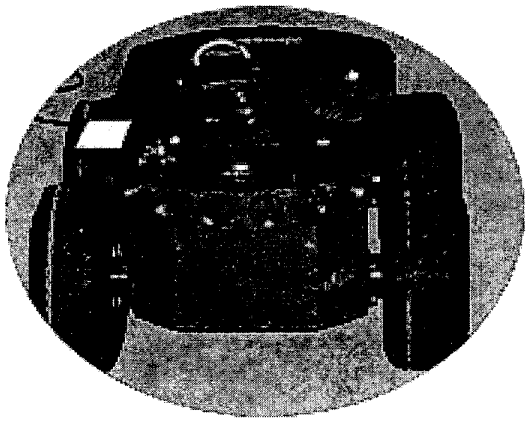

Figure 7: Pioneer II robot.

configurations are trained and tested from a room with different obstacle locations to a hallway next to the lab. Similar values for γ and α to the previous example are used. In the case of a real robot, vision algorithms are used to detect the searched object, in this case another robot. Figure 8 shows the hallway problem while Figure 9 displays a room with obstacles, both described in the RAVE environment.

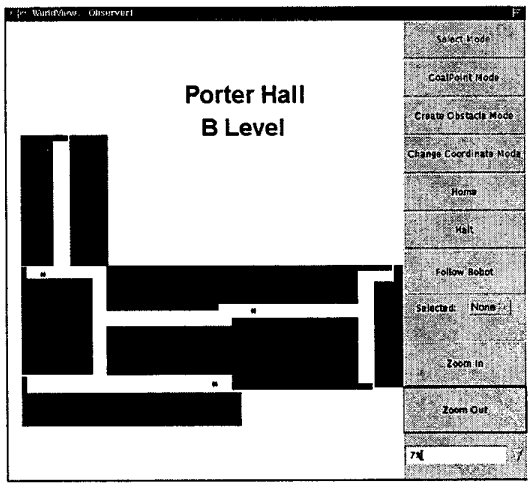

Figure 8: Hallway environment in RAVE observer window.

Figure 9 shows a possible set of trajectories established by the robots after training. It should be noted that the robots performed different trajectories over time. This dynamic variation may result in greater adaptability to changing environments.

This approach to the problem constitutes an alternative to the usual planning techniques for mobile robots that involve the design of check points, where robots have to rotate in order to cover an area. It should also be noted the increase in the information exchange between robots needed to be able to formulate the problem in this context. It may constitute a drawback, but in most cases the extra location information is usually transmitted between robots for other purposes.

The combination of RAVE, PB3A and the Q-learning algorithm allows the system to derive the team strategy in simulation, faster than real-time, and switched to the real robots once training is done.

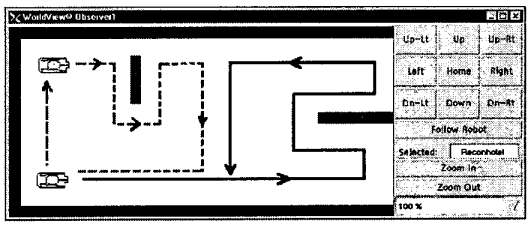

Figure 9: Trajectories determined by the Q-learning algorithm for 2 robots in a room environment with sparse obstacles.

4 Summary

A decentralized Q-learning algorithm to compute distributed value functions is analyzed. It is shown that the Q-learning algorithm has a solution that is a weighted sum of future discounted rewards over all the agents. Two examples are shown: urban traffic control systems and mobile robot planning task. Its power comes from the capability to partition the larger problem into smaller ones but still being able to make local decisions with a global perspective. Parameter selection and convergence issues need to be addressed. This approach is specially suitable for cooperation between teams of similar agents. For heterogeneous systems, hierarchical architectures need to be studied. The dynamics of the adaptability of the planning trajectories of the mobile robots generated by this approach needs to be studied. Further developments for the traffic control example will include the use of vehicle-to-vehicle communication, inclusion of bus schedules, emergency vehicle activity, and vehicle navigation features.

Acknowledgements

We would like to thank K. Dixon, J. Jackson, R. Malak, Y. Patel and T. Pham for the development of the PB3A architecture, D. Manstetten and T. Schwab, Bosch, for their help with the traffic simulator ARTIST and Thomas Ke, Trans Associates, Inc., for providing the traffic data for Penn-Circle. This work is supported in part by DARPA/ETO under contracts F30602-96-2-0240 and DABT63-97-1-0003 and by the Institute for Complex Engineered Systems at Carnegie Mellon University. We also thank the Intel Corporation for providing part of the computing hardware.

References

[1] J.J. Cheese, M. Cartwright, I.W. Routledge and B. Radia. UTMC- the UK initiative for ITS: An update: Results of the first call, progress of the second call. In *Proceedings of the Intn'l Conference on Road Transportation, Information and Control*, pages 100–103, London, England, April 1998.

[2] K.R. Dixon, J.M. Dolan, W.S. Huang, C.J.J. Paredis and P.K. Khosla. RAVE: A real and virtual environment for multiple robot systems. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Volume 3, pages 1360–67, October 1999.

[3] K.R. Dixon, T.Q. Pham and P.K. Khosla. Port-based adaptable agent architecture. In *Proceedings of the International Workshop on Self-adaptive Software*, Oxford University, England, April 2000.

[4] H. Fujii, O. Hayashi and N. Nakagata. Experimental research on inter-vehicle communication using infrared rays. In *Proceedings of the 1996 IEEE Intelligent Vehicles Symposium*, pages 266–71, Tokyo, Japan, September 1996.

[5] W. Krautter, T. Bleile, D. Manstetten and T. Schwab. Traffic simulation with ARTIST. In *Proceedings of Conference on Intelligent Transportation Systems*, pages 472–77, Boston, MA, Nov 1997.

[6] J. Schneider, W.K. Wong, A. Moore and M. Riedmiller. Distributed value functions. In *Machine Learning. Proceedings of the Sixteenth International Conference on Machine Learning*, pages 371–8, Bled, Slovenia, June 1999.

[7] E. Sourour and M. Nakagawa. Mutual decentralized synchronization for intervehicle communications. *IEEE Transactions on Vehicular Technology*, Volume 48, Number 6, pages 2015–27, 1999.

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