TECHNICAL RESEARCH REPORT

A Distributed Algorithm for Solving a Class of Multi-agent Markov Decision Problems

by Hyeong Soo Chang, Michael C. Fu

TR 2003-25

A Distributed Algorithm for Solving a Class of Multi-agent Markov Decision Processes

Hyeong Soo Chang and Michael C. Fu*

February 27, 2003

Abstract

We consider a class of infinite horizon Markov decision processes (MDPs) with multiple decision makers, called agents, and a general joint reward structure, but a special decomposable state/action structure such that each individual agent's actions affect the system's state transitions independently from the actions of all other agents. We introduce the concept of "localization," where each agent need only consider a "local" MDP defined on its own state and action spaces. Based on this localization concept, we propose an iterative distributed algorithm that emulates gradient ascent and which converges to a locally optimal solution for the average reward case. The solution is an "autonomous" joint policy such that each agent's action is based on only its local state. Finally, we discuss the implication of the localization concept for discounted reward problems.

Keywords: multi-agent, Markov decision process, distributed algorithm, local optimal solution, stochastic control

*H.S. Chang can be reached by email at hschang@ccs.sogang.ac.kr and is with the Department of Computer Science and Engineering, Sogang University, Seoul, Korea, and M.C. Fu can be reached by telephone at +1 (301) 405-2241, by fax at +1 (707) 897-3774, or by email at mfu@rhsmith.umd.edu and is with the Robert H. Smith School of Business and the Institute for Systems Research, University of Maryland, College Park, MD 20742-1871, U.S.A.

1 Introduction

We consider a class of decision-making problems under uncertainty where there are multiple decision makers, called agents, which act independently of each other, but have a common goal of maximizing a system-wide reward function. We model these systems as multi-agent, discrete-time Markov decision processes (MDPs) with a general joint reward structure, but a special *product-form* state transition probability function that "factors" into the product of functions, each of which depends only on localized state/action information of an individual agent.

As a result, aside from the reward structure, the original MDP is decomposable into smaller "local" MDPs based only on actions and states of a particular agent. However, actually "solving" such a local MDP (i.e., finding an optimal local policy) requires the specification of a reward function that depends only on local state/action information, but the overall (global) reward function itself is not assumed separable. Thus, it is not obvious a priori how to exploit the decomposition appropriately, because each agent needs to take an action in its local state in a "cooperative" manner to maximize the global reward.

On the other hand, even when the local state and action spaces of the agents are relatively small, the global MDP may still suffer from the *curse of the dimensionality*, making solution of the global MDP using standard approaches, such as policy or value iteration, computationally intractable. A natural approach is to try to find a solution scheme where certain tasks are distributed to the individual agents, and then the results for the tasks are "merged" or "coordinated" via some predetermined protocol of information communication among the agents, and this general process repeated until a given terminating condition is satisfied. However, such a distributed approach should meet the following conditions to be sound and "efficient". First, the overhead of the communication among the agents must not be large. Second, the tasks assigned to each agent need to be small in terms of time and space complexities. Finally, the merged results from the tasks of each agent must provide a "useful" and "meaningful" global solution. To the best of the authors' knowledge, there is no existing work that satisfies these conditions for the setting we consider, where there is a joint reward structure. This paper is a step toward developing an efficient distributed control scheme that meets the above conditions when the local state and action spaces of each agent are relatively small.

Our "localization" of the original MDP into local MDPs for each individual agent converts the joint reward function into a local reward function dependent only on an individual agent's states and actions by projecting with respect to the stationary distributions of the Markov chains induced by the policies of the other agents. We show that solving the local MDP for a given agent provides an optimal "reactive" local policy, given any fixed set of local policies of the other agents. Based on this, we present an iterative distributed algorithm that converges to a locally optimal solution of the global MDP, where the solution is an overall policy such that each agent's decision is based on

only its own local state, i.e., each agent need not observe the states of the other agents, eliminating any communication overhead among the agents in implementing the policy.

The algorithm starts from an arbitrary local policy for each agent. Given the currently selected policies of the other agents and the corresponding induced stationary distributions, each agent computes its best local reactive policy by solving its current local MDP. During the computation of the best autonomous local reactive policy, there is no communication among the agents. With a certain monotonicity property, the algorithm converges to a local optimal solution for the global MDP.

Some previous work on decentralized control of finite-state Markov processes [12] [1] considered partitioned state and/or action spaces with a delayed sharing information structure. Even though an autonomous joint policy structure is considered, the algorithms for computing an optimal policy with such structure are centralized with the given global parameters, and thus there is no reduction in dimensionality when carrying out the solution procedure. Similarly, asynchronous implementations of value iteration (see, e.g., [19]) presented in [13] [4] are done with the global parameters via a communication protocol. Kushner and Chen [16] present an algorithm that generates a set of independent "local" subproblems at each iteration, which can be solved in parallel, by the Dantzig-Wolfe decomposition technique based on a linear programming formulation for a given MDP, where the MDP does not have a partitioned state/action space. However, the algorithm is based on the assumption that grouping of states is possible such that each group is connected with other groups via some border states that do not belong to any group and need to be identified in advance. The performance of the algorithm depends on how such grouping is done. Each independent subproblem for each group is defined with a "local" parameter of the group, but at each iteration, a centralized problem must be solved using the solutions to the local subproblems.

This paper is organized as follows. In Section 2, we define the class of MDPs considered in this paper and present some examples that can be modeled by such MDPs. In Section 3, we introduce the concept of the localization, and in Section 4, we provide an iterative distributed algorithm for average reward problems based on the localization concept. We discuss the implication of localization for discounted reward problems in Section 5. Section 6 concludes with some remarks.

2 Multi-agent Markov Decision Processes

2.1 Model

We formally describe the class of MDPs with N multiple decision makers, called agents, that we consider. Each agent $i = 1, ..., N < \infty$ has its own *finite* state space X_i and *finite* action space A_i. For simplicity, it is assumed that every action in A_i is admissible at each state in X_i for agent i. We define a *local policy* $\pi_i : X_1 \times \cdots \times X_N \to A_i$ for agent i and denote the set of all possible such

local policies for agent i as Π_i. We also define an *autonomous local policy* $\phi_i : X_i \to A_i$ for agent i, and denote the set of all possible such local policies for agent i as Φ_i. Thus, an autonomous local policy prescribes an action from the agent's local action set depending on only the agent's local state.

Each agent i is associated with a *local* state transition function $P_i : X_i \times A_i \to \mathcal{D}(X_i)$, where $\mathcal{D}$ signifies a probability distribution over X_i. Denote the probability of transitioning from state $x_i \in X_i$ to $y_i \in X_i$ by taking action $a_i \in A_i$ at x_i by $P_i(y_i|x_i, a_i)$. Given a local reward function $R_i : X_i \times A_i \to \mathcal{R}$, denote the *local* MDP for the agent i as $M_i = (X_i, A_i, R_i, P_i)$.

Define a *global* MDP $M = (X, A, P, R)$ from the dynamics of each agent as follows: the joint state space $X = X_1 \times \cdots \times X_N$ so that a state $x \in X$ is an N-tuple $x = (x_1, ..., x_N)$; the joint action space $A = A_1 \times \cdots \times A_N$ so that an action $a \in A$ is an N-tuple $a = (a_1, ..., a_N)$; the joint state transition probability is *product form*, i.e.,

$$P(y|x, a) = \prod_{i=1}^{N} P_i(y_i|x_i, a_i), \quad x, y \in X, a \in A;$$

and the joint reward function is general $R : X \times A \to \mathcal{R}$, i.e., unlike the state transition function, which is the product of the local transition functions, there is a priori no relationship assumed between the global reward function R and local reward functions R_i. We assume that R is bounded such that there exists a constant $M < \infty$ such that $\max_{x,a} R(x, a) \leq M$, and also assume that there is no interdependent constraint on the action set, i.e., an action taken by agent i does not affect or limit the feasible action space $A_j, j \neq i$, of any other agent.

Solving each local MDP $M_i, i = 1, ..., N$ independently will not necessarily lead to a solution of the global MDP, since the local reward functions R_i and the global reward function R are not assumed related. We denote a joint policy for all the agents as an N-tuple $\pi = (\pi_1, ..., \pi_N)$ and Π as the set of all possible such joint policies, and we say that a policy π is *fully decentralized* if $\pi = (\phi_1, ..., \phi_N)$ with $\phi_i \in \Phi_i, i = 1, ..., N$. We will make the following assumption throughout.

Assumption 2.1 *For each $i = 1, ..., N$, the local MDP M_i is unichain.*

Note that the property of an MDP being unichain is completely unrelated to the reward function. Furthermore, under the above assumption, the global MDP M is unichain, too.

There are several (ergodicity) conditions from which we can check whether or not an MDP is unichain (e.g., [11, p.56]). Perhaps the simplest condition, called the "minorant" condition, is that there exists a state reachable (with positive probability) from every other state. For example, the system can reach a "reset" state with a positive probability (from any state with any action). Suppose that a given MDP is not unichain. We can add then an artificial state $\bar{x} = (\bar{x}_1, ..., \bar{x}_N)$ to X by adding $\bar{x}_i$ to $X_i, i = 1, ..., N$ such that $\bar{x}_i$ is reachable from any local state x_i with probability (w.p.) $\epsilon \approx 0$ by taking any admissible action a at x. But to make each agent not willing to reach

the added local state in their optimal decisions, we make the immediate reward of taking any action at $\bar{x}_i$ extremely small. In this way, we can transform the given MDP into a unichain MDP, and an optimal solution for the unichain MDP can approximate an optimal solution for the original MDP very closely.

For $\pi \in \Pi$ and $x \in X$, define the infinite horizon average reward value function:

$$J^{\pi}(x) = \lim_{H \to \infty} \frac{E\left\{\sum_{t=0}^{H-1} R(Y_t, \pi(Y_t)) | Y_0 = x\right\}}{H},$$

where $\{Y_t, t = 0, 1, ...\}$ denotes the state of the MDP at time t. It is well-known that because the global MDP is unichain, $J^{\pi}(x)$ is independent of the initial state x (e.g., [19]), and we use g^{π} to denote the corresponding infinite horizon average reward of following the policy π. A (joint) policy $\pi^* \in \Pi$ is *optimal* if it achieves the maximum average reward g^*, i.e.,

$$g^{\pi^*} = g^* \geq g^{\pi} \text{ for any } \pi \in \Pi.$$

The time-complexity of solving the global MDP M is $O(|X|^2|A|)$ for carrying out one policy improvement step in policy iteration or one iteration in value iteration [17]. Therefore, if either $|X|$ or $|A|$ is large, solving the MDP M via the well-known exact methods is impractical, motivating the need for good decentralized policies that can be found efficiently. In general, there may not exist a fully decentralized policy that is optimal, but in the setting considered in this paper, we provide an algorithm that will find a full decentralized policy that achieves a form of local optimality. Each agent i considers its own local MDP knowing the currently selected local policies of all other agents and the corresponding induced stationary distributions.

2.2 Some examples

In this section, we provide a couple of examples that have the special structure on the state transition function that we assume in the setting of this paper. Other examples that we do not describe here, but that fall into our framework, include the air vehicle battle management problems described in the recent work by Arslan et al. [3], and certain classes of distributed database problems.

2.2.1 Multi-robot navigation

Consider multiple "robots" operating on a common area, such as a two-dimensional rectangular grid, where the overriding objective is to navigate them such that they are kept at a safe distance apart. Alternatively, the objective might be to bring them all together. During each time step, each of the robots chooses independently to stay put or attempt to make a move of one unit in any permissible direction, but there would be some random mechanism determining whether the chosen move would actually be executed. For example, there could be a Bernoulli probability that the move is successful, and otherwise either the robot stays put or with some (e.g., equal) probability

moves to another neighboring state. The overall reward function to be maximized would be some function of the Euclidean distances between the current location of the robots, for example the minimum distance or the average (pairwise) distance. If the goal were to bring the robots together, then a similar cost function (e.g., maximum distance) could be minimized.

2.2.2 Admission control

Queueing networks are commonly used to model communication networks [2] and manufacturing systems. An important class of problems involves admission control into the system. We describe some simple examples in this class that fall into our framework.

There are several, say $K > 1$, independent sources of jobs (e.g., different traffic streams on the Internet) that feed one or several first-come, first-served (FCFS) queues. The actions involve admission control at each of these sources. The number of jobs that the server can process in a time period may also be stochastic, e.g., due to breakdowns or other unplanned events or randomness in the system. The global objective must tradeoff between throughput and queue length (or equivalently, delay), and clearly depends jointly on each source. In our framework, the MDP model would specify an agent for each of the different arrival sources and possibly one agent for the server. There are myriad variations that have this basic structure. We will describe two of these in more detail now.

In the first variation, there is a separate upstream queue for each of the K independent sources, and the decision to be made is whether to admit or to reject a newly arriving job into the queue. The job arrival process for each queue is described by a Markov Modulated Bernoulli Process (MMBP) [10] such that for the queue i, there is a finite number of traffic states in the set S_i and the transition dynamics between the traffic states is governed by a Markov chain. Denote $\delta^i_{s_i,s'_i}$ as the probability of transitioning from state $s_i \in S_i$ to s'_i for the MMBP model of the job arrival process at queue i. At each state $s_i \in S_i$, one job is generated w.p. $\lambda_i(s_i) > 0$. Once a job is admitted into queue i, each job departs queue i w.p. μ_i (i.i.d.) after staying at least one unit decision time. The departed jobs from each upstream queue are fed into a downstream FCFS single-server queue, which for this variation is assumed to have infinite buffer capacity and serves one job each period (if there is one to be processed). An additional agent is required to model this server, though in this variation the server has no action choices; thus, this MDP model has $K + 1$ agents.

A local state x_i for an agent corresponding to queue $i = 1, ..., K$, is (n_i, s_i, ϕ_i), where n_i is the number of the current jobs (already admitted) at the (upstream) station i; s_i is the current traffic state of the MMBP model for the job arrival process; $\phi_i \in \{0, 1\}$, where 0 indicates a new job arrival and 1 indicates no new job arrival. A local action a_i is either 1 for accepting a job or 0 for rejecting a job. For the server, the local state x_{K+1} is simply given by the number of jobs in its buffer, and

the transition to the next local state is given (deterministically) by $(x_{K+1} + \sum_{i=1}^{K} d_i - 1)^+$, where d_i represents the number of jobs that departed from queue i and $x^+ = \max(x, 0)$.

The global transition structure is product form. For $x_i = (n_i, s_i, \phi_i)$ and $y_i = (n_i', s_i', \phi_i')$, if $n_i' = n_i + a_i\phi_i - d_i$, then

$$P_i(y_i|x_i, a_i) = \delta^i_{s_i,s_i'}[\lambda_i(s_i')\phi_i' + (1 - \lambda_i(s_i'))(1 - \phi_i')](\mu_i)^{d_i}, \quad d_i = 1, 2, ..., n_i,$$

and 0 otherwise. Then, we have

$$P(y|x, a) = \prod_{i=1}^{K} P_i(y_i|x_i, a_i)$$

for $y_{K+1} = (x_{K+1} + \sum_{i=1}^{K} d_i - 1)^+$, and 0 otherwise.

The global immediate (random) reward function could take the following general form:

$$R(x, a) = \sum_{i=1}^{K} a_i - \zeta \left(x_{K+1} + \sum_{i=1}^{K} n_i \right),$$

where $\zeta > 0$ is a tradeoff parameter between throughput and queue length.

In the second variation, again assume that each of the underlying arrival processes is Markov modulated, with the same notation for the underlying Markov chain. This time, however, the capacity of the queue is finite, and the action is a choice of the number of jobs to generate from a set dependent on the local Markov state, i.e., at each state x_i, source i can generate $a_i \in \{0, ..., \alpha_i(x_i)\}$ jobs with $\alpha_i(x_i) < \infty$. In addition, the server chooses how many jobs to attempt to process, with a success (all or none of the jobs served) probability dependent on the number of jobs attempted. Thus, this MDP model again has $K + 1$ agents.

Denote the queue capacity by $M < \infty$. If Q is the queue length at the beginning of the period, a_i is the number of jobs generated by source i in the current period, and $\eta \leq Q$ is the number of jobs processed in the period, then the queue length at the beginning of the next period is given by the following expression:

$$\min \left\{ Q + \sum_{i=1}^{K} a_i, M \right\} - \eta. \tag{1}$$

Note that if the number of newly generated jobs exceeds the queue capacity M, then some decision rule must be invoked in order to discard the excess jobs, but since the job characteristics are all i.i.d., the decision rule in our simplest setup does not matter.

Assume that all jobs are identical and a job can be processed during one time period, starting with the period *after* its arrival. The stochastic service mechanism is such that if there are $Q > 0$ jobs in the queue, a server attempting to process $i \in \{1, ..., Q\}$ jobs is successful w.p. $\gamma^Q(i)$, $\sum_{i=1}^{Q} \gamma^Q(i) = 1$. Again, failure means that all jobs stay in the queue for an additional period. In other words, in (1), $\eta = i$ w.p. $\gamma^Q(i)$ and 0 w.p. $1 - \gamma^Q(i)$. If $Q = 0$, the server is idle and takes no action.

The global MDP state $x = (x_1, ..., x_K, x_{K+1})$ is defined by the Markov chain state for each source $x_i \in S_i,\ i = 1, ..., K$, along with the number of the pending jobs in the queue x_{K+1}. The global action $a = (a_1, ..., a_K, a_{K+1})$ is defined by the number of jobs generated at each source, along with the number of jobs attempted to be processed by the server $a_{K+1} \in \{0, 1, ..., \min(x_{K+1}, M)\}$.

Again, it is not difficult to see that the transition dynamics are of product form. From (1) and the service mechanism,

$$P(y|x, a) = \gamma^{x_{K+1}}(a_{K+1}) \prod_{i=1}^{K} \delta^i_{x_i, y_i},$$

for $y_{K+1} = \min(x_{K+1} + \sum_{i=1}^{K} a_i, M) - a_{K+1}$,

$$P(y|x, a) = (1 - \gamma^{x_{K+1}}(a_{K+1})) \prod_{i=1}^{K} \delta^i_{x_i, y_i},$$

for $y_{K+1} = \min(x_{K+1} + \sum_{i=1}^{K} a_i, M)$ (and 0 otherwise).

The immediate (average) reward function R is given by

$$R(x, a) = \gamma^{x_{K+1}}(a_{K+1}) \cdot a_{K+1} - \zeta \cdot \left(\min \left\{ x_{K+1} + \sum_{i=1}^{K} a_i, M \right\} - a_{K+1} \gamma^{x_{K+1}}(a_{K+1}) \right),$$

where the expectation w.r.t. the server success probability has been taken, and again $\zeta > 0$ is a tradeoff parameter between throughput and queue length.

We can easily extend the above examples to a *load balancing* problem, where there are $\kappa > 1$ parallel servers, leading to a model with $K + \kappa$ agents, where agent $K + 1, ..., K + \kappa$ represents the server at each queue. Agents 1 through K, representing the job arrival sources, not only need to control the sending rate but also need to determine where to dispatch the generated jobs among the κ servers. This decision certainly depends on the current load of each queue and the service characteristic of each queue.

3 Localization

Suppose that all of the agents except one, say agent i, have fixed autonomous local policies, $\{\phi_j \in \Phi_j : j \neq i\}$, leaving just the choice of local policy for agent i to be determined. Define the following average reward value-like function, which is a constant independent of the initial state:

$$g_i^* := \max_{\pi_i \in \Pi_i} \lim_{H \to \infty} \frac{1}{H} E \left\{ \sum_{t=0}^{H-1} R\left(x_t, (\phi_1(x_t^1), ..., \pi_i(x_t), ..., \phi_N(x_t^N))\right) \middle| x_0 \in X \right\},$$

where $x_t = (x_t^1, ..., x_t^N)$ and $x_t^j \in X_j$ denotes the local state at time t for agent $j = 1, ..., N$. Note that g_i^* is defined over *all* local policies for agent i, and not just autonomous local policies. In this section, we show using the concept of localization that g_i^* can be achieved using an *autonomous* local policy by defining the reward function R_i appropriately.

From standard MDP theory, constraining the action choices for agent $j \neq i, j = 1, ..., N$, by the autonomous local policies ϕ_j, under Assumption 2.1, there exists a bounded function h_i defined over X that satisfies the following equation: for any $x = (x_1, ..., x_N) \in X$,

$$
\begin{aligned}
& g_i^* + h_i((x_1, ..., x_N)) = \\
& \quad \max_{a_i \in A_i} \Bigg(R\left(x, (\phi_1(x_1), ..., a_i, ..., \phi_N(x_N))\right) \\
& \quad + \sum_{y_k \in X_k, k=1,...,N} \left[P_i(y_i|x_i, a_i) \prod_{j \neq i, j=1,...,N} P_j(y_j|x_j, \phi_j(x_j)) \right] h_i((y_1, ..., y_N)) \Bigg). \qquad (2)
\end{aligned}
$$

Thus, for any $a_i \in A_i$,

$$
\begin{aligned}
& g_i^* + h_i((x_1, ..., x_N)) \geq \\
& \quad R\left(x, (\phi_1(x_1), ..., a_i, ..., \phi_N(x_N))\right) \\
& \quad + \sum_{y_k \in X_k, k=1,...,N} \left[P_i(y_i|x_i, a_i) \prod_{j \neq i, j=1,...,N} P_j(y_j|x_j, \phi_j(x_j)) \right] h_i((y_1, ..., y_N)). \qquad (3)
\end{aligned}
$$

Let $\{\rho_j(x), x \in X_j\}$ denote the stationary distribution of the Markov chain induced on the local state space of agent j by autonomous local policy ϕ_j. Then, summing up both sides of Equation (**??**) w.r.t. $\rho_j, j \neq i$, gives (where the notation $\sum_{x_{j \neq i} \in X_j}$ is shorthand for the double summation $\sum_{j=1,...,N; j \neq i} \sum_{x_j \in X_j}$)

$$
\begin{aligned}
& g_i^* + \sum_{x_{j \neq i} \in X_j} \left[\prod_{j \neq i} \rho_j(x_j) \right] h_i((x_1, ..., x_N)) \\
& \geq \sum_{x_{j \neq i} \in X_j} \prod_{j \neq i} \rho_j(x_j) \Bigg(R\left(x, (\phi_1(x_1), ..., a_i, ..., \phi_N(x_N))\right) \\
& \quad + \sum_{y_k \in X_k, k=1,...,N} \left[P_i(y_i|x_i, a_i) \prod_{j \neq i, j=1,...,N} P_j(y_j|x_j, \phi_j(x_j)) \right] h_i((y_1, ..., y_N)) \Bigg) \\
& \geq \sum_{x_{j \neq i} \in X_j} \prod_{j \neq i} \rho_j(x_j) R\left(x, (\phi_1(x_1), ..., a_i, ..., \phi_N(x_N))\right) \\
& \quad + \sum_{y_1, ..., y_i, ..., y_N} \left(\left[\sum_{x_{j \neq i}} \prod_{j \neq i} [\rho_j(x_j) P_j(y_j|x_j, \phi_j(x_j))] \right] P_i(y_i|x_i, a_i) \right) h_i((y_1, ..., y_N)) \\
& = \sum_{x_{j \neq i}} \prod_{j \neq i} \rho_j(x_j) R\left(x, (\phi_1(x_1), ..., a_i, ..., \phi_N(x_N))\right) \\
& \quad + \sum_{y_i \in X_i} P_i(y_i|x_i, a_i(x_i)) \left(\sum_{y_{j \neq i} \in X_j} \left[\prod_{j \neq i} \rho_j(y_j) \right] h_i((y_1, ..., y_N)) \right), \qquad (4)
\end{aligned}
$$

where the last step follows from the invariance property of the stationary distribution (see, e.g., [11, p.57]): for any $\phi_k \in \Phi_k$,

$$\sum_{x_k \in X_k} \rho_k(x_k) P_k(y_k|x_k, \phi_k(x_k)) = \rho_k(y_k).$$

Define an *average* value function defined over X_i from h_i or "projection" of h_i w.r.t. the stationary distributions ρ_j over X_j of the selected policies ϕ_j from the other agents $j \neq i$:

$$\bar{h}_i(x_i) := \sum_{x_{j\neq i} \in X_j} \left[\prod_{j \neq i} \rho_j(x_j) \right] h_i((x_1, ..., x_N)), x_i \in X_i. \tag{5}$$

Similarly, define also an *average* reward function w.r.t. ϕ_j:

$$\bar{R}_i(x_i, a_i) := \sum_{x_{j\neq i} \in X_j} \left[\prod_{j \neq i} \rho_j(x_j) \right] R\left(x, (\phi_1(x_1), ..., a_i, ..., \phi_N(x_N))\right), \quad x_i \in X_i, a_i \in A_i. \tag{6}$$

Since Equation (3) holds for any $a_i \in A_i$, it holds in particular for the action that maximizes the right-hand side of (3), so

$$g_i^* + \bar{h}_i(x_i) \geq \max_{a_i \in A_i} \left(\bar{R}_i(x_i, a_i) + \sum_{y_i \in X_i} P_i(y_i|x_i, a_i) \bar{h}_i(y_i) \right), \quad x_i \in X_i.$$

Denoting an action that achieves the right hand side of Equation (2) as a_i^*, and summing up both sides over the stationary distributions of $\rho_j, j \neq i$, Equation (2) can be rewritten as

$$g_i^* + \bar{h}_i(x_i) = \bar{R}_i(x_i, a_i^*) + \sum_{y_i \in X_i} P_i(y_i|x_i, a_i^*) \bar{h}_i(y_i), \quad x_i \in X_i.$$

It trivially follows that

$$g_i^* + \bar{h}_i(x_i) \leq \max_{a_i \in A_i} \left(\bar{R}_i(x_i, a_i) + \sum_{y_i \in X_i} P_i(y_i|x_i, a_i) \bar{h}_i(y_i) \right), \quad x_i \in X_i.$$

Therefore,

$$g_i^* + \bar{h}_i(x_i) = \max_{a_i \in A_i} \left(\bar{R}_i(x_i, a_i) + \sum_{y_i \in X_i} P_i(y_i|x_i, a_i) \bar{h}_i(y_i) \right), \quad x_i \in X_i, \tag{7}$$

which we refer to as a *localized* version of Bellman's optimality equation, since the equation involves only X_i and A_i.

Now consider an MDP with X_i, A_i, P_i, and $R_i = \bar{R}_i$. Under Assumption 2.1, there exists a constant κ and a bounded function ζ defined over X_i for which

$$\kappa + \zeta(x_i) = \max_{a_i \in A_i} \left(R_i(x_i, a_i) + \sum_{y_i \in X_i} P_i(y_i|x_i, a_i) \zeta(y_i) \right), \quad x_i \in X_i, \tag{8}$$

and if a constant κ' and a function ζ' satisfy Equation (7), then $\kappa' = \kappa$ (see Theorem 8.4.3 in [19]). Notice that g_i^* and $\bar{h}_i$ satisfy Equation (7).

We summarize the localization concept below as a theorem:

Theorem 3.1 *Given $\phi_j \in \Phi_j$ and corresponding $\{\rho_j(\cdot)\}$ for all agents $j \neq i$, under Assumption 2.1, for $M_i = (X_i, A_i, P_i, R_i)$ with $R_i = \bar{R}_i$, there exists a bounded function ζ defined over X_i and a constant κ such that*

$$\kappa + \zeta(x_i) = \max_{a_i \in A_i} \left(R_i(x_i, a_i) + \sum_{y_i \in X_i} P_i(y_i|x_i, a_i)\zeta(y_i) \right), \quad x_i \in X_i, \tag{9}$$

and any policy $\phi \in \Phi_i$ which, for each local state $x_i \in X_i$, prescribes an action that maximizes the right hand side of Equation (8) achieves g_i^* with $\kappa = g_i^*$.

We remark that if we add the condition that $\sum_{x_i \in X_i} \rho_i(x_i)\zeta(x_i) = 0$ to Equation (8), where ρ_i is the stationary distribution of the Markov chain induced from such a policy ϕ for agent i as defined by the theorem, then $\zeta = \bar{h}_i$ from the uniqueness of ζ. As a special case of $\bar{R}_i$, if each agent k is associated with its local (bounded) reward function $R_k : X_k \times A_k \to \mathcal{R}$ and $R(x, a) = \sum_k R_k(x_k, a_k)$, then $\bar{R}_i(x_i, a_i) = R_i(x_i, a_i) + \sum_{j \neq i, j=1,...,N} \psi^{\phi_j}$, where ψ^{ϕ_j} is the average reward of following the policy ϕ_j w.r.t. $M_j = (X_j, A_j, P_j, R_j)$. Note that if there is no constraint on the action choices across the agents, for this case just solving the local MDP independently and forming a composite global policy provides an optimal joint policy for the global MDP. This particular case gives an intuitive argument for the results of the localization concept.

The localization theorem result is intuitively reasonable but also somewhat surprising. If an agent needs to maximize the global average reward constrained for a given set of fixed autonomous local policies of the other agents, and if the joint state transition structure is product form, maximizing the local average reward, defined with the projected reward function w.r.t. the stationary distributions of the fixed autonomous local policies of the other agents, is equivalent to maximizing the (constrained) original global average reward. Furthermore, there exists an *autonomous* local policy for agent i that achieves this maximal reward g_i^*. In other words, one might expect that in order to achieve g_i^*, a local policy would depend on the states of the other agents. However, by the above theorem, the agent need only consider its own local state.

The localization theorem result can be extended to *Borel* state spaces as long as Equation (2) can be stated with the existence of the constant g_i^* and the function h_i and the existence of the stationary distributions for the selected policies $\phi_j \in \Phi_j$. A simple condition for this being true is that each local MDP $M_i, i = 1, ..., N$, satisfy one of the recurrent conditions given in [11, p.57].

Iterative solution methods to obtain κ and ϕ_i follow directly from the well-known average reward value iteration and policy iteration procedures. We briefly review policy iteration. Agent i starts with an arbitrary policy $\phi_i^0 \in \Phi_i$ and iterates the following steps: at iteration $n \geq 0$, agent i obtains $\psi^{\phi_i^n}$ and ζ^n that satisfy the following:

$$\psi^{\phi_i^n} + \zeta^n(x_i) = \bar{R}_i(x_i, \phi_i^n(x_i)) + \sum_{y_i \in X_i} P_i(y_i|x_i, \phi_i^n(x_i))\zeta^n(y_i), x_i \in X_i,$$

where this step is called *policy evaluation*. Then a policy ϕ_i^{n+1} such that

$$\phi_i^{n+1}(x_i) \in \arg\max_{a_i \in A_i} \left(\bar{R}_i(x_i, a_i) + \sum_{y_i \in X_i} P_i(y_i|x_i, a_i)\zeta^n(y_i) \right), x_i \in X_i,$$

is obtained, where this step is called *policy improvement*. Eventually, ϕ_i^n converge to an optimal (in the sense of Theorem 3.1) ϕ_i within a finite number of iterations. The localization naturally induces a simple distributed iterative algorithm that converges to a local optimal solution for the global MDP M.

4 A Distributed Algorithm for a Local Optimal Solution

We describe the algorithm in a constructive way, rather than giving a pseudocode for it, and for the two-agents case ($N = 2$) for simplicity. Extending the algorithm for $N > 2$ is straightforward.

Assume that the global joint reward function R is known to both agents. Each agent 1 and 2 starts with its own initial policy $\alpha^0 \in \Phi_1$ and $\beta^0 \in \Phi_2$. At iteration $k \geq 1$, agent 1(2) informs agent 2(1) of $\alpha^{k-1}(\beta^{k-1})$ and the corresponding stationary distribution $\rho_1^{k-1}(\rho_2^{k-1})$. Then agent 1 solves the local MDP $M_1^k = (X_1, A_1, P_1, R_1^k)$, where $R_1^k(x_1, a_1) = \sum_{x_2 \in X_2} \rho_2^{k-1}(x_2) R(x_1, x_2, a_1, \beta^{k-1}(x_2))$. An optimal policy for M_1^k is α^k. Similarly, agent 2 solves the local MDP $M_2^k = (X_2, A_2, P_2, R_2^k)$, where $R_2^k(x_2, a_2) = \sum_{x_1 \in X_1} \rho_1^{k-1}(x_1) R(x_1, x_2, \alpha^{k-1}(x_1), a_2)$, to obtain β^k. Because at each iteration, each agent finds the best autonomous local policy with respect to the policy of the other agent, as the gradient is the direction of the greatest local increase in a given objective function, the underlying idea is similar to that of the gradient-ascent algorithm.

By the above construction, we first have the following fact: for $k \geq 2$ and $k = 2m$ with $m = 1, 2, ...,$

$$g^{(\alpha_k, \beta_{k-1})} \geq g^{(\alpha_{k-2}, \beta_{k-1})} \geq g^{(\alpha_{k-2}, \beta_{k-3})} \geq g^{(\alpha_{k-4}, \beta_{k-3})} \geq \cdots \geq g^{(\alpha_2, \beta_1)} \geq g^{(\alpha_0, \beta_1)}.$$

Similarly,

$$g^{(\alpha_{k-1}, \beta_k)} \geq g^{(\alpha_{k-1}, \beta_{k-2})} \geq g^{(\alpha_{k-3}, \beta_{k-2})} \geq g^{(\alpha_{k-3}, \beta_{k-4})} \geq \cdots \geq g^{(\alpha_1, \beta_2)} \geq g^{(\alpha_1, \beta_0)}.$$

That is, the performances of the pairs of the policies of the agent 1 and 2 monotonically improve in a zigzag manner across the agents' views. We state this property as a proposition.

Proposition 4.1 *For $k \geq 3$ and $k = 2m$ with $m = 1, 2, ...,$ the following monotonicity holds:*

$$g^{(\alpha_k, \beta_{k-1})} \geq g^{(\alpha_{k-2}, \beta_{k-3})} \text{ and } g^{(\alpha_{k-1}, \beta_k)} \geq g^{(\alpha_{k-3}, \beta_{k-2})}.$$

Recall that α_k achieves $g_1^* = \max_{\pi_1 \in \Pi_1} g^{(\pi_1, \beta_{k-1})}$ and β_k achieves $g_2^* = \max_{\pi_2 \in \Pi_2} g^{(\alpha_{k-1}, \pi_2)}$ for $k \geq 1$. Therefore, we have that for $k \geq 1$,

$$g^{(\alpha_k, \beta_{k-1})} \geq g^{(\pi_1, \beta_{k-1})} \text{ for any } \pi_1 \in \Pi_1$$

and for $k \geq 2$,

$$g^{(\alpha_{k-2},\beta_{k-1})} \geq g^{(\alpha_{k-2},\pi_2)} \text{ for any } \pi_2 \in \Pi_2.$$

It follows that we can state the following result:

Proposition 4.2 *For $k \geq 2$ and $k = 2m$ with $m = 1, 2, ...,$*

$$g^{(\alpha_k,\beta_{k-1})} \geq \begin{cases} g^{(\pi_1,\beta_{k-1})} \text{ for any } \pi_1 \in \Pi_1 \\ g^{(\alpha_{k-2},\pi_2)} \text{ for any } \pi_2 \in \Pi_2 \\ g^{(\pi_1,\beta_{k-3})} \text{ for any } \pi_1 \in \Pi_1 \\ g^{(\alpha_{k-4},\pi_2)} \text{ for any } \pi_2 \in \Pi_2 \\ \cdots \\ g^{(\alpha_0,\pi_2)} \text{ for any } \pi_2 \in \Pi_2 \end{cases}$$

Similarly, for $k \geq 2$ and $k = 2m$ with $m = 1, 2, ...,$

$$g^{(\alpha_{k-1},\beta_k)} \geq \begin{cases} g^{(\alpha_{k-1},\pi_2)} \text{ for any } \pi_2 \in \Pi_2 \\ g^{(\pi_1,\beta_{k-2})} \text{ for any } \pi_1 \in \Pi_1 \\ g^{(\alpha_{k-3},\pi_2)} \text{ for any } \pi_2 \in \Pi_2 \\ g^{(\pi_1,\beta_{k-4})} \text{ for any } \pi_1 \in \Pi_1 \\ \cdots \\ g^{(\pi_1,\beta_0)} \text{ for any } \pi_1 \in \Pi_1 \end{cases}$$

It is then immediately true that in the best case, at the kth iteration with $k \geq 2$ and $k = 2m, m = 1, 2, ...,$ $O\left(k(|A_1|^{|X|} + |A_2|^{|X|})\right)$ suboptimal policies are eliminated from Π.

At each iteration $k \geq 3$, agent 1 checks whether or not

$$g^{(\alpha_k,\beta_{k-1})} = g^{(\alpha_{k-2},\beta_{k-3})},$$

and agent 2 checks if

$$g^{(\alpha_{k-1},\beta_k)} = g^{(\alpha_{k-3},\beta_{k-2})}.$$

If both of the conditions hold, then the algorithm stops. Otherwise, both agents continue their communications and (distributed) computations. The communication requires that each agent pass to the other agent the currently computed best reactive autonomous local policy and the corresponding stationary distribution, along with a flag indicating the status of its stopping condition. Because of the monotonicity and the finite number of the (joint) policies in Π, both conditions will hold eventually after some finite number of iterations.

Even if both of the conditions are true, this does not imply that the performance of the converged policy sets of agents 1 and 2 are equal, nor does it ensure that either one of them has found an optimal joint policy for the global MDP. It only indicates that the algorithm has converged to locally optimal policies, and we select the policy set with the greater performance.

Suppose that $|A_i| = C$ and $|X_i| = D$ for all $i = 1, ..., N$. As discussed earlier, the time complexity of applying just one "policy improvement" step of policy iteration to solve the global MDP is $O(C^N \cdot D^{2N})$. The policy iteration algorithm converges to an optimal joint policy in $O(C^{N^{D^{2N}}})$ iterations in the worst case, making the total worst-case time complexity of solving the global MDP $O(C^N \cdot D^{2N} \cdot C^{N^{D^{2N}}})$. For our algorithm, if each agent applies the policy iteration algorithm in parallel to its local MDP, solving the local MDP takes $O(C^D D^2)$ in the worst case with the communication cost of $O(D)$ and with acomputation cost $O(C \cdot D^N)$ of the average reward function. If our algorithm converged to an optimal joint policy for the global MDP in the worst case, it would have taken at most $O(C^D \cdot D^2 \cdot C^{N^{D^{2N}}})$ iterations neglecting the computational cost of the average reward function. Therefore, the *worst-case* time complexity of solving the global MDP in a global manner can be exponentially alleviated by our algorithm depending on the size of C, D and N if the average reward function can be computed efficiently. Thus, an efficient implementation of the projection operation for the average reward function is key to the practical effectiveness of our algorithm. One possible way to address the computational problem of the average reward function is to use a sample mean, where the sampling is done over the stationary distributions so that more emphasis is given to frequently visited states.

Even though our algorithm converges to only a locally optimal solution in theory, the algorithm may serve as a good heuristic. There are several published works that a policy obtained from one-step policy improvement with a "good" heuristic policy is near-optimal for various problems (see, e.g., [18] [8] [15] [22] [6] [14]). Similarly, each agent can start with a good heuristic policy available for its local MDP or the given global MDP and apply some number of iterations of the proposed algorithm to generate an improved joint policy. Note that in contrast to one-step policy improvement, in the best case, each agent will eliminate an exponential number of suboptimal policies.

5 Implication on Discounted Rewards

For the discounted reward case, we define the following value function ($\pi \in \Pi, x \in X$):

$$V^\pi(x) = E\left\{\sum_{t=0}^{\infty} \gamma^t R(x_t, \pi(x_t)) \middle| x_0 = x\right\}, 0 < \gamma < 1,$$

and refer to $V^*(x) = \max_{\pi \in \Pi} V^\pi(x)$ as the optimal value at x for the infinite horizon discounted reward criterion. The nominal goal is again to find an optimal joint policy $\pi^* \in \Pi$ that achieves the optimal value.

Given an agent i and for a set of fixed policies $\{\phi_j \in \Phi_j : j \neq i\}$ from the other agents, we define a value function V_i^* over X such that

$$V_i^*((x_1, ..., x_N)) := \max_{\pi_i \in \Pi_i} E\left\{\sum_{t=0}^{\infty} \gamma^t R\left(x_t, (\phi_1(x_t^1), ..., \pi_i(x_t), ..., \phi_N(x_t^N))\right) \middle| x_0 = (x_1, ..., x_N) \in X\right\},$$

where $x_t = (x_t^1, ..., x_t^N)$ and $x_t^j \in X_J$ denotes the local state at time t for agent j. Define a function $\bar{V}_i^*$ such that

$$\bar{V}_i^*(x_i) := \sum_{x_{j\neq i} \in X_j, j=1,...,N} \left[\prod_{j\neq i, j=1,...,N} \rho_j(x_j) \right] V_i^*((x_1, ..., x_N)), x_i \in X_i.$$

As before, $\bar{V}_i^*$ is an *average* value function defined over X_i from V_i^* or projection of V_i^* w.r.t. the stationary distributions ρ_j over X_j of the selected policies ϕ_j from the other agents $j \neq i$.

With similar arguments as in the average reward case for Belleman's optimality equation of the infinite horizon discounted reward criterion, it can be proven that the following result holds.

Theorem 5.1 *Given $\phi_j \in \Phi_j$ and corresponding $\{\rho_j(\cdot)\}$ for all agents $j \neq i$, under Assumption 2.1, for $M_i = (X_i, A_i, P_i, R_i)$ with $R_i = \bar{R}_i$, there exists a bounded function V defined over X_i such that*

$$V(x_i) = \max_{a_i \in A_i} \left(R_i(x_i, a_i) + \gamma \sum_{y_i \in X_i} P_i(y_i | x_i, a_i) V(y_i) \right), x_i \in X_i, \tag{10}$$

and any policy $\phi_i \in \Phi_i$ which, for each local state $x_i \in X_i$, prescribes an action which maximizes the right hand side of Equation (9) achieves $\bar{V}_i^*(x_i)$.

As we can see from the theorem, the projected value at a particular agent i's local state of the optimal value function for the global MDP w.r.t. the stationary distributions of the other agents' autonomous local policies is equal to the optimal value function value at the particular agent i's local state for the local MDP defined with the projected reward function w.r.t. the stationary distributions of the other agents' autonomous local policies. However, unlike the average reward case, an autonomous local policy that achieves the V-function value does not necessarily achieve the V_i^*-function value. It merely achieves the projected value of the V_i^*-function value. This is because for the infinite horizon discounted reward criterion, the optimal action choice for an agent to optimize the global reward function depends on the given *initial* local states of the other agents.

Certainly, we can develop an iterated algorithm as in the average reward case by using the localization result given in Theorem 5.1. In this case, each agent will obtain the best autonomous reactive policy that achieves the optimal projected value function with respect to the autonomous policies chosen by the other agents.

Recently, there has been a lot of work directed at approximating the (optimal) value function with a set of linearly parameterized functions or "basis functions" (see, e.g., [5]), where the V^*-function is approximated by a parameterized function

$$\tilde{V}(x, r) = \sum_{k=1}^{K} r_k \zeta_k(x), r \in \mathcal{R}^K, \zeta_i : X \to \mathcal{R}, i = 1, ..., K.$$

Researchers have concentrated on how to compute the value of the weight vector r to "fit" $\tilde{V}(x,r)$ to $V^*(x)$ under the assumption that the functions $\{\zeta_i(x)\}$ are given (see, e.g., [9]), but it is very difficult to design such basis functions. This is because there is no general rule to extract some "features" contained in a given problem. Our localization result provides a general rule to select a set of basis functions. We can use $\{\bar{V}_i^*(x_i), i = 1, ..., N\}$ as a set of basis functions such that

$$\tilde{V}((x_1, ..., x_N), r) = \sum_{k=1}^{N} r_k \bar{V}_k^*(x_k) \text{ with } \sum_{k=1}^{N} r_k = 1, r_k \geq 0 \text{ for all } k = 1, ..., N.$$

6 Concluding Remarks

The local optimality result of the proposed algorithm can be enhanced by introducing a random restart, which is commonly used in several global optimum seeking algorithms. We can generate several initial local policies at random and apply the algorithm, or we can generate random local policies once the algorithm converges to a local optimal solution.

At each iteration of the algorithm, each agent has a set of autonomous local policies of all agents. If desired, once we apply the proposed algorithm for a certain number of iterations, we can stop the algorithm and can generate a global policy, combining each set of local policies of the agents via methods called "parallel rollout" or "policy switching" [8, 7], which improves the performances of all joint policies from each set.

If the computation of the stationary distribution for each agent is impractical, we can approximate the stationary distribution by several methods [21], leading to an approximate version of our algorithm.

References

[1] M. Aicardi, F. Davoli, and R. Minciardi, "Decentralized optimal control of Markov chains with a common past information set," *IEEE Trans. Automat. Control*, AC-32, pp. 1028–1031, 1987.

[2] E. Altman, "Applications of Markov decision processes in communication networks : a survey," *Markov Decision Processes, Models, Methods, Directions, and Open Problems*, E. Feinberg and A. Shwartz (Eds.) Kluwer, pp. 488-536, 2001.

[3] G. Arslan, J. D. Wolfe, J. Shamma, and J. L. Speyer, "Optimal planning for autonomous air vehicle battle management," in *Proc. of the 41st IEEE CDC*, 2002, pp. 3782–3787.

[4] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods.* Englewood Cliffs, NJ: Prentice-Hall, 1989.

[5] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming.* Athena Scientific, 1996.

[6] S. Bhulai and G. Koole, "On the structure of value functions for threshold policies in queueing models," Technical Report 2001-4, Department of Stochastics, Vrije Universiteit Amsterdam, 2001.

[7] H. S. Chang, *On-line sampling-based control for network queueing problems*, Ph.D. Thesis, School of Electrical and Computer Engineering, Purdue University, 2001.

[8] H. S. Chang, R. Givan, and E. K. P. Chong, "Parallel rollout for on-line solution of partially observable Markov decision processes," *Discrete Event Dyn. Syst.,* 2002, revised.

[9] D. P. de Farias and B. Van Roy, "The linear programming approach to approximate dynamic programming," to appear in *Operations Research.*

[10] W. Fischer and K. Meier-Hellstern, "The Markov-modulated Poisson process (MMPP) cookbook," *Performance Evaluation*, vol. 18, pp. 149–171, 1992.

[11] O. Hernández-Lerma, *Adaptive Markov Control Processes.* Springer-Verlag, 1989.

[12] K. Hsu and S. I. Marcus, "Decentralized control of finite state Markov processes," *IEEE Trans. Automat. Control*, AC-27, pp. 426–431, 1982.

[13] A. Jalali and M. J. Ferguson, "On distributed dynamic programming," *IEEE Trans. Automat. Control*, vol. 37, no. 5, pp. 685–689, 1992.

[14] A. Kolarov and J. Hui, "On computing Markov decision theory-based cost for routing in circuit-switched broadband networks," *J. Network and Systems Management,* vol. 3, pp. 405-425, 1995.

[15] G. Koole and Philippe Nain, "On the value function of a priority queue with an application to a controlled pollying model," *Queueing Systems*, vol. 34, pp. 199–214, 2000.

[16] H. J. Kushner and C. Chen, "Decomposition of systems governed by Markov chains," *IEEE Trans. Automat. Control*, AC-19, no. 5, pp. 501–507, 1974.

[17] M. Littman, T. Dean, and L. Kaelbling, "On the complexity of solving Markov decision problems," in *Proc. 11th Annual Conf. on Uncertainty in Artificial Intelligence*, 1995, pp. 394–402.

[18] T. J. Ott and K. R. Krishnan, "Separable routing: a scheme for state-dependent routing of circuit switched telephone traffic," *Ann. Oper. Res.* vol. 35, pp. 43–68, 1992.

[19] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* Wiley, New York, 1994.

[20] S. M. Ross, *Applied Probability Models with Optimization Applications.* San Francisco, CA: Holden-Day, 1970.

[21] P. J. Schweitzer, "A survey of aggregation-disaggregation in large Markov chains," in *Proc. 1st Int. Workshop on the Numerical Solution of Markov Chains*, 1990.

[22] N. Secomandi, "Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands," *Comput. Oper. Res.,* vol. 27, pp. 1201–1225, 2000.