Localization and a Distributed Local Optimal Solution Algorithm for a Class of Multi-Agent Markov Decision Processes

Hyeong Soo Chang

Abstract: We consider discrete-time *factorial* Markov Decision Processes (MDPs) in multiple decision-makers environment for infinite horizon average reward criterion with a general joint reward structure but a factorial joint state transition structure. We introduce the "localization" concept that a global MDP is localized for each agent such that each agent needs to consider a local MDP defined only with its own state and action spaces. Based on that, we present a gradient-ascent like iterative distributed algorithm that converges to a local optimal solution of the global MDP. The solution is an *autonomous* joint policy in that each agent's decision is based on only its local state.

Keywords: Distributed algorithm, local optimal solution, Markov decision process, multi-agent system, stochastic control.

1. INTRODUCTION

We consider discrete-time *factorial* Markov Decision Processes (MDPs) in multiple decision-makers environment for infinite horizon average reward criterion with a general joint reward structure but a factorial joint state transition structure such that each agent makes his state transitions independently from the other agents according to his own local state transition structure. Even though the *global* MDP is factorial, solving each *local* MDP for each agent independently does not necessarily provide a global MDP solution due to the general reward structure. Each agent needs to take action at its local state in a cooperative manner to maximize the global average reward due to the reward structure.

Unfortunately, the complexity of solving the global MDP is often large even if the local state and action spaces of each agent are small. To break *the curse of dimensionality*, an approach is to develop a cooperative solution scheme whereby certain tasks are distributed to the agents, and the results of the tasks are *merged* or *coordinated* via some predetermined communication protocol. Then, new tasks are distributed to the agents and the same processes continue until a certain 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 be kept to a minimum. Second, the task that has been assigned to each agent needs to be small in terms of time and space complexities. Finally the merged result must provide a useful and meaningful global solution. However, the nature of the interdependencies among the agents makes it very difficult to design such a scheme and to the best of the author's knowledge, there are few prior works on developing any distributed algorithm with these aspects satisfied. The present paper is a step toward developing an efficient distributed control scheme that meets the above conditions under the assumption that the local state and action spaces of each agent are small.

We introduce the concept of "localization" whereby a factorial MDP is localized for each agent such that each agent needs to consider a local MDP defined only with its own state and action spaces or its local parameters. Given a selected local policy for each agent, the global MDP is *projected* into a local MDP for each agent with respect to the selected policies of the other agents. The projection is performed by the stationary distributions of the Markov chains induced from the policies of the other agents. We will show that solving the local MDP for an agent provides the best *reactive* policy to the policies of the other agents. Based on this, we present an iterative distributed algorithm that converges to a local optimal solution of the global MDP. The solution is an *autonomous* joint policy where each agent's decision is based only on its local state. This policy will be useful in that during the actual policy invocation over the decision making process, each agent does not need to observe the states of the other agents.

The algorithm starts with an arbitrary local policy

Manuscript received November 11, 2002; revised March 14, 2003; accepted May 27, 2003. This research was supported by Institute for Applied Science and Technology at Sogang University, Seoul, Korea.

Hyeong Soo Chang is with the Department of Computer Science and Engineering, Sogang University, Seoul, Korea (e-mail: hschang@sogang.ac.kr).

for each agent. At each iteration of the algorithm, every agent needs only the information of the flag for a certain termination condition, the currently selected local policies of the other agents, and the stationary distributions of the local policies of the other agents for their currently localized MDPs.

With that information, each agent computes the best local reactive policy with respect to the local policies of the other agents by solving its currently localized MDP. The computation of the best local reactive policy does not require any communication among the agents. With a certain monotonicity property, the algorithm converges to a local optimal solution for the global MDP.

Some previous works on the decentralized control of finite-state Markov processes [1, 11] have 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. Similarly, asynchronous implementations of value iteration (see, e.g., [18]) presented in [4, 12] are done with the global parameters via a communication protocol. Kushner and Chen [15] 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 in a topological sense. Each independent sub-problem for each group is defined with a local parameter of the group, but at each iteration, a centralized problem must be solved with solutions to the local sub-problems.

This paper is organized as follows. In Section 2, we describe a class of MDPs we consider and provide some motivating example problems with such structures of MDPs. In Section 3, we then introduce the concept of the localization and in Section 4, we provide an iterative distributed algorithm based on the localization concept. We conclude the paper in Section 5 with some remarks.

2. MULTI-AGENT MARKOV DECISION PROCESSES

2.1. Model

We formally describe a class of Markov decision processes with multiple decision makers that we consider in the present paper.

Each decision maker or agent $i=1,\ldots,N<\infty$ has its own *finite* state space X_i and *finite* action space A_i. It is assumed that every action in A_i is admissible at each state in X_i for agent i for simplicity. We define a *local policy* $\theta_i : X_1 \times \cdots \times X_N \to A_i$ for agent i and denote the set of all possible such local policies for agent i as θ_i and also define an *autonomous local policy* $\pi_i : X_i \to A_i$ for agent i and denote the set of all possible such local policies for agent i as Π_i.

Each agent i is associated with its *local* state transition function P_i such that $P_i : X_i \times A_i \to D(X_i)$, where D signifies a probability distribution over X_i. We denote the probability of transitioning from state $x_i \in X_i$ to $y_i \in X_i$ by taking an action $a_i \in A_i$ at x_i as $P(y_i \mid x_i, a_i)$. Given a local reward function R_i such that $R_i : X_i \times A_i \to \Re$, we denote the *local* MDP for agent i as $M_i = (X_i, A_i, R_i, P_i)$.

We define a *global* MDP $M = (X, A, P, R)$ as follows from the dynamics of each agent: the joint state space X is given such that $x \in X$ is an N-tuple $x = (x_1, \ldots, x_N)$, the joint action space A such that $a \in A$ is an N-tuple $a = (a_1, \ldots, a_N)$, the joint state transition probability such that for $x, y \in X$ and $a \in A$, $P(y \mid x, a) = \prod_{i=1}^{N} P_i(y_i \mid x_i, a_i)$, and a joint reward function $R : X \times A \to \Re$. We assume that R is bounded such that there exists a constant $M < \infty$ with $\max_{x,a} R(x,a) \le M$ and also assume that there is no interdependent constraint on the action set. That is, taking an action a_i by agent i does not affect or limit the action choice of any other agent.

From the global state transition function, we consider the class of MDPs that are *factorial*. However, solving each local MDP $M_i, i=1,\ldots,N$ independently does not provide the solution of the global MDP necessarily due to the general structure of the global reward function. This is because even though the global MDP is factorial, each agent needs to take an action at its local state in a cooperative manner in order to maximize the global average reward due to the reward structure.

We denote a joint policy for all the agents as an N-tuple $\theta = (\theta_1, \ldots, \theta_N)$ and Θ as the set of all possible such joint policies, and we state that a policy θ is *fully decentralized* if $\theta = (\pi_1, \ldots, \pi_N)$ with $\pi_i \in \Pi_i, i=1,\ldots,N$. We will make the following assumption throughout the present paper:

Assumption 1: For each $i=1,\ldots,N$, the local MDP M_i is unichain.

Note that the unichainedness is independent of the local reward function, and by the above assumption, the global MDP M is also unichain.

Given $\theta \in \Theta$, define the following value

function $J^{\theta}(x) = \lim_{H\to\infty} \frac{E\left\{\sum_{t=0}^{H-1} R(x_t, \theta(x_t)) \mid x_0 = x\right\}}{H}$,

where H is the horizon size and x_t is a random variable that denotes a state in X at time t. It is well-known that under Assumption 1, $J^{\theta}(x)$ is independent of the initial state x (see, e.g., [18]) and is the same for all states. We denote the average reward of following the policy θ as a constant g^{θ}.

The goal is to find an optimal joint policy $\theta^* \in \Theta$ that achieves the *optimal* average reward g^* that satisfies

$$g^* = g^{\theta^*} \geq g^{\theta} \text{ for any } \theta \in \Theta.$$

The time-complexity of solving the global MDP M is $O(|X|^2|A|)$ for applying just one "policy improvement" step in policy iteration and for just one iteration in value iteration [16]. Therefore, if either $|X|$ or $|A|$ is large, solving the MDP M via the well-known exact methods is impractical.

2.2. Some remarks on the model

Some problems are involved with an exogenous stochasticity - a stochastic component that does not depend on control actions taken by the agents. For example, for robot navigation, certain unknown things can obstruct the robot from carrying out proper navigation where the occurrences of the events are independent of the navigational actions of the robot.

This component can be added to our model just as an *artificial* agent that makes independent transitions from the other agents with a proper definition of its action space. The artificial agent always undertakes one action of "do" at every local state.

There are several (ergodic) conditions from which we can check the unichainedness of MDPs. See for example [10, p. 56]. The simplest condition, called the "minorant" condition, is where exists a state, call it 0, such that for any pair of a state and an admissible action at that state, with a positive probability, 0 is reachable from that 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 then add an artificial state $\bar{x} = (\bar{x}_1, \ldots, \bar{x}_N)$ to X by adding $\bar{x}_i$ to $X_i, i = 1, \ldots, N$ such that $\bar{x}_i$ is reachable from any local state x_i with probability $\varepsilon \approx 0$ by taking any admissible action a at x. But to make each agent unwilling 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 unichained MDP and an optimal solution for the unichained MDP can approximate an optimal solution for the given MDP very closely.

3. SOME FACTORIAL MDPS

In this section, we provide simple examples that have the factorial structure we imposed on the MDP model to motivate the localization concept approach. We do not discuss the unichainedness property here as we provided the related remarks in the previous section.

3.1. Multi-robot navigation

Assume that there are $K > 1$ robots, in a $B \times B$ grid world, where a robot's position is described with two-tuple (h, v) with $h, v \in G = \{1, \ldots B\}$. A robot's local state is its position expressed by a two-tuple, and its local action is to choose from among left, right, up, down, and stay. If a robot chooses left in a state (h, v), then with probability $p > 0$, it moves to $(h-1, v)$ if $(h-1, v) \in G \times G$. It moves to one of the *reachable* neighbors with the equal probability $q > 0$ such that the sum of probabilities is one and becomes stuck in the current position with probability one if $(h-1, v) \notin G \times G$, where all of the moves are made independently from the other robots. We can define the robot's stochastic position change for other actions (right, up, and down) similarly. If a robot chooses the stay action, the robot stays at its current position.

The immediate joint cost from particular moving actions taken by each robot is the maximum value of (average) mutual Euclidean distances among the robots. The goal is that each robot (starting from a distance away from the other robots) needs to navigate the grid world to minimize the joint cost function, that is, all of the robots wish to "meet" together at a position by minimizing the average mutual distances.

3.2. Job sending rate control

It is very important to design an *elastic* application that has the ability to adapt to the system performance. For example, in the case of the Internet, an application that sends the video packets must adapt the sending rate depending on the network conditions [2] Or, in a certain manufacturing system, a machine that produces a certain part must adapt its production rate depending on the production rates of other machines in order to increase the overall production utility.

To model this into a decision making problem, consider a single FIFO queue Q and its single server that processes a certain number of jobs stochastically at a time (e.g., due to the load caused by other demands, the failure or the idleness of the server machine, etc.), where we assume that all jobs have the same processing time as the unit decision time, and further assume that the queue has a finite capacity I so that at most I jobs can be found in the queue. If there are $B > 0$ number of jobs in the queue, the

server processes $b \in \{1,\ldots,B\}$ jobs with $\gamma^B(b)$ positive success probability with $\sum_{b=1}^{B}\gamma^B(b)=1$. If case of failure, all jobs remain in the queue. If $B=0$, the server does not do anything.

There are $K>1$ applications that generate jobs and the newly generated jobs from the applications are merged into the single queue Q. Each application i generates (independently from the other applications) jobs depending on its local state s_i in a finite set S_i. We assume that there is a Markov chain that describes the transition dynamics among states in S_i and denotes $\delta^i_{s_i,s_i'}$ as the transition probability from state $s_i \in S_i$ to $s_i' \in S_i$. At each state s_i, it can generate $\Omega_i \in \{0,\ldots,\alpha_i\}$ jobs with $\alpha_i < \infty$.

Let $\Omega_{i,t}$ be the number of jobs generated from the application i at time interval $(t,t+1)$, and let B_t be the current queue size of Q, and let η_t be the number of jobs processed over $(t,t+1)$. The queue size (before task(s) is (are) processed) dynamics is then expressed by the following equation:

$$B_{t+1} = \max\left\{\min\left\{B_t + \sum_{i=1}^{K}\Omega_{i,t}, I\right\} - n_t, 0\right\}.$$

Note that to apply the minimization operation in the right side of the above equation, the server at the queue Q must discard a certain number of jobs if adding the newly generated jobs overloads the queue. We assume that there exists a fixed rule to achieve this. The event sequence is as follows. The queue starts in an empty state. Each application generates jobs and the jobs arrive into the queue. The server performs overload control by discarding some jobs, if necessary. The server then processes the jobs (stochastically).

We view this system as $(k+1)$- agents system with K applications and one server at the queue. State $x=(x_1,\ldots,x_K,x_{K+1})$ is given such that for $i=1,\ldots,K$, $x_i = s_i \in S_i$ and $x_{K+1}=B$, where B is the number of the pending jobs in the queue. Action $a=(a_1,\ldots,a_K,a_{K+1})$ is given such that for $i=1,\ldots,K$, $a_i=\Omega_i$ and $a_{K+1}\in\{0,1,\ldots,I\}$ is the number of jobs to be processed by the server. Note that not all actions can be taken. The admissibility depends on x_{K+1}. It is straightforward to incorporate an admissible action set shown in the model we discussed in Section 1.

Each job generation state transition dynamics is independent of its local action and the next x'_{K+1} local state is determined once the action $a_1,\ldots,a_K$ is fixed. Furthermore, given a state x and an action a taken at x, the two local states for the server are reachable. For $a_{K+1}>0$, x_{K+1} can reach x'_{K+1} when the server processes a job successfully or $x''_{K+1}=x_{K+1}$ when the server fails. Therefore, the state transition is defined such that for all y with success,

$$P(y\mid x,a) = \prod_{i=1}^{K}\delta^i_{s_i,s_i'}\gamma^{x_{K+1}}(a_{K+1})$$

and for all y with failure,

$$P(y\mid x,a) = \prod_{i=1}^{K}\delta^i_{s_i,s_i'}(1-\gamma^{x_{K+1}}(a_{K+1})).$$

The case for $a_{K+1}=0$ is similarly defined. The agents need to cooperate with each other to maximize the average throughput, the average number of successfully processed jobs, while simultaneously minimizing the average queue size. It is obvious that to achieve high throughput and low queue size, each application must control the sending rate by proper selection of the number of jobs to be processed. To express the competing objectives into one reward function, we introduce a trade-off parameter $\lambda>0$ that represents a relative importance between the throughput and the queue size. The (average immediate) reward function R is given such that

$$R(x,a) =$$

$$\gamma^{x_{K+1}}(a_{K+1})\left(a_{K+1} - \lambda\left(\min\left\{x_{K+1}+\sum_{i=1}^{K}a_i, I\right\} - a_{K+1}\right)\right)$$

$$-\lambda(1-\lambda^{x_{K+1}}(a_{K+1}))\times\left(\min\left\{x_{K+1}+\sum_{i=1}^{K}a_i, I\right\} - a_{K+1}\right)$$

$$\text{if } \min\left\{x_{K+1}+\sum_{i=1}^{K}a, I\right\} - a_{K+1} > 0, \text{ and}$$

$$\gamma^{x_{K+1}}(a_{K+1})a_{K+1} \text{ if}$$

$$\min\left\{x_{K+1}+\sum_{i=1}^{K}a, I\right\} - a_{K+1} > 0, \text{ and 0 otherwise.}$$

We can simply extend the above example to a *load balancing* problem. Consider now $\kappa>1$ parallel queues, making $K+\kappa$ agents with agents $K+1,\ldots,K+\kappa$ being the server at each queue. The agents 1 through K, or applications, not only need to control the sending rate but also need to determine where to dispatch the generated jobs among the queues. This decision certainly depends on the current load of each queue and the service characteristic of each queue.

3.3. Distributed database

Consider a set of simple searching systems or database $S_i=\{s_1^i,\ldots,s_m^i\}$ with $m<\infty$ for searching agent $i=1,\ldots,K<\infty$. One component for a local

state for agent i is $s_i \in S_i$. If a query q_i in a finite set Q_i is imposed, agent i must determine whether to process the query at the server s_i or to pass the query to the other servers in S_i. Once agent i makes the determination to pass the processing to a particular server in S_i, the query can be passed into the particular server deterministically or probabilistically (it stays at the current server with a probability). We assume that a document or necessary information for each query term $q_i \in Q_i$ can be searched independently of each other. For example, $q = (q_1, q_2, q_3)$ with $K = 3$ can be given such that q_1 is a term for searching an image, q_2 for a sound, and q_3 for a text to locate a relevant document.

The local state of agent i is given as $x_i =< s_i, q_i >, s_i \in S_i$ and $q_i \in Q_i$ and local action set A_i for agent i is S_i. A global reward function is given such that $R(x,a)$ stands for "relevance" factor of the selected servers to a query $q = (q_1, \ldots, q_K)$ in a coherent manner. For example, the relevance factor can be related with the "query term frequency" (the number of occurrences of the terms in a query q in a document that a server indexes) with a hitting factor that reflects the degree of matches between the given query and the outputs of the searching process from the selected servers. That is, even though the searching process of each agent is independent, the reward function expresses the joint relevance of the jointly selected servers to the query q. For example, if the query <Markov, decision, process> is given, once particular servers are determined, the agent 1 will search for the relevant documents to "Markov" and the agent 2 will do the same for "decision" and the agent 3 will do the same for "process". However, each agent needs to select its local server in a cooperative manner with the other agents such that the overall outputs from the search are relevant to the "Markov decision process".

For our problem example, we assume that each query $q_i \in Q_i, i = 1, \ldots, K$ is generated with probability $\lambda_i(q_i) > 0$ with $\sum_{q_i \in Q_i} \lambda_i(q_i) = 1$, where the value of $\lambda_i(q_i)$ is estimated from "historical" usage statistics, from which we further assume that the query q_i is generated independently from the query $q_{j \neq i}$. Each Q_i contains the *empty query* and with the empty query, an agent must remain at the current server.

With the problem description as above, the global transition structure is factorial and each agent needs to cooperate in order to maximize the global reward function.

3.4. Admission control in multi-stations

Consider parallel multiple stations that need to perform admission control for incoming calls to each station. Each station $i = 1, 2, \ldots, K < \infty$ is associated with an infinite buffer and is required to decide whether to admit or to reject a newly arriving call into the buffer. The call arrival process for each station is described by the Markov Modulated Bernoulli Process (MMBP) [9] such that for the station 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. We 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 call arrival process at the station i. At each state $s_i \in S_i$, one call is generated with probability $\lambda_i(s_i) > 0$. Once a call is admitted into the station i, each call departs the station i with the same probability μ_i after remaining for at least one unit of decision time.

The departed call(s) from each (upstream) station is (are) fed into a single infinite FIFO (downstream) queue. If multiple calls are fed into the queue at the same time, we assume that the calls from the station i are queued before the calls from the station j if $i < j$. The calls in the downstream queue are routed out to certain network such that one call from each station, if any, is routed out deterministically in FIFO manner. The setup of the problem here can serve as a model for systems arising in production networks, communication networks, etc.

Local state x_i for the agent in the station i is $(n_i, s_i, \phi_i, \eta_i)$, where n_i is the number of the current calls (that are already admitted) at the (upstream) station i, and s_i is the current traffic state of the MMBP model for the call arrival process, and $\phi_i \in \{0,1\}$ with 0 being a new arrival and 1 being no arrival, and η_i is the number of the current calls at the (downstream) queue from the station i. Local action a_i is either 1 for accept or 0 for reject.

The global transition structure is factorial. For $x_i = (n_i, s_i, \phi_i, \eta_i)$ and $y_i = (n_i', s_i', \phi_i', \eta_i')$, if $n_i' = n_i + a_i\phi_i - d$ and $\eta_i' = \max\{\eta_i + d - 1, 0\}$ for $d = 1, 2, \ldots, n_i$, then

$$P_i(y_i \mid x_i, a_i) = \delta^i_{s_i, s_i'}[\lambda_i(s_i')\phi_i' + (1 - \lambda_i(s_i'))(1 - \phi_i')](\mu_i)^d$$

and 0 otherwise. The global (average) immediate reward function is given such that

$$R(x,a) = f\left(\sum_{i=1}^{K} a_i\right) - \zeta \cdot g\left(\sum_{y \in X} P(y \mid x, a)\left(\sum_{i=1}^{K} \eta_i'\right)\right),$$

where f is an increasing positive function (the more accepted calls, the more rewards) that represents a global throughput utility, and g is an increasing positive function that represents a global queue size cost,

and $\varsigma > 0$ is a trade-off parameter between the throughput utility and the queue size utility.

Each agent is required to cooperate in order to maximize the average throughput utility and to minimize the average queue size utility with a given trade-off parameter ς at the single FIFO queue.

4. LOCALIZATION OF GLOBAL MDP

It is intuitively true that if all of the agents except one select their policies and follow them, optimizing the global average reward with those fixed policies must be able to be concentrated on only when the maximizing agent has not fixed its own policy. However, it is not trivial to show formally that this is the case. In this section, we show this by the concept of localization.

Given an agent i and for a set of selected autonomous local policies $\{\pi_j \in \Pi_j : j \neq i\}$ from the other agents $j \neq i$, we define a constant g_i^* such that given $x_0 \in X$,

$$g_i^* := \max_{\theta_i \in \Theta_i} \lim_{H \to \infty} \frac{1}{H} E\left\{ \sum_{t=0}^{H-1} R\left(x_t, (\pi_1(x_t^1), \ldots, \theta_i(x_t), \ldots, \pi_N(x_t^N))\right)\right\},$$

where $x_t = (x_t^1, \ldots, x_t^N)$ and $x_t^j \in X_j$ denotes the local state at time t for agent $j = 1, \ldots N$.

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

$$\begin{aligned} & g_i^* + h_i((x_1, \ldots, x_N)) = \\ & \max_{a_i \in A_i} R\left(x, (\pi_1(x_1), \ldots, a_i, \ldots, \pi_N(x_N))\right) \\ & + \sum_{y_k \in X_k, k=1,\ldots,N} \left[P_i(y_i \mid x_i, a_i) \times \right. \\ & \left. \prod_{j \neq i, j=1,\ldots N} P_j(y_j \mid x_j, \pi_j(x_j))\right] h_i((y_1, \ldots, y_N)). \end{aligned} \tag{1}$$

Trivially, for any $a_i \in A_i$,

$$\begin{aligned} & g_i^* + h_i((x_1, \ldots, x_N)) \geq \\ & R\left(x, (\pi_1(x_1), \ldots, a_i, \ldots, \pi_N(x_N))\right) \\ & + \sum_{y_k \in X_k, k=1,\ldots,N} \left[P_i(y_i \mid x_i, a_i) \times \right. \\ & \left. \prod_{j \neq i, j=1,\ldots N} P_j(y_j \mid x_j, \pi_j(x_j))\right] h_i((y_1, \ldots, y_N)). \end{aligned} \tag{2}$$

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 π_j. Then, summing up both sides of Equation (2) with respect to $\rho_j j \neq i$, gives

$$\begin{aligned} & g_i^* + \sum_{x_{j \neq i} \in X_j} \left[\prod_{j \neq i} \rho_j(x_j)\right] h_i((x_1, \ldots, x_N)) \\ & \geq \sum_{x_{j \neq i} \in X_j} \prod_{j \neq i} \rho_j(x_j) \left(R\left(x, (\pi_1(x_1), \ldots, a_i, \ldots, \pi_N(x_N))\right)\right. \\ & + \sum_{y_k \in X_k, k=1,\ldots,N} \left[P_i(y_i \mid x_i, a_i) \times \right. \\ & \left.\left. \prod_{j \neq i, j=1,\ldots N} P_j(y_j \mid x_j, \pi_j(x_j))\right] h_i((y_1, \ldots, y_N))\right) \\ & \geq \sum_{x_{j \neq i} \in X_j} \prod_{j \neq i} \rho_j(x_j) \left(R\left(x, (\pi_1(x_1), \ldots, a_i, \ldots, \pi_N(x_N))\right)\right. \\ & + \sum_{y_1, \ldots, y_i, \ldots, y_N} \left(\left[\sum_{x_j \neq i} \prod_{j \neq i} [\rho_j(x_j) P_j(y_j \mid x_j, \pi_j(x_j))]\right]\right. \\ & \left.\left. \times P_i(y_i \mid x_i, a_i)\right) h_i((y_1, \ldots, y_N))\right) \\ & = \sum_{x_{j \neq i}} \prod_{j \neq i} \rho_j(x_j) \left(R\left(x, (\pi_1(x_1), \ldots, a_i, \ldots, \pi_N(x_N))\right)\right. \\ & + \sum_{y_i \in X_i} P_j(y_j \mid x_j, \pi_j(x_j)) \times \\ & \left.\left(\sum_{y_{j \neq i} \in X_j} \left[\prod_{j \neq i} \rho_j(y_j)\right] h_i((y_1, \ldots, y_N))\right)\right), \end{aligned} \tag{3}$$

where the last step follows from the invariance property of the stationary distribution (see, e.g., [10, pg. 57]): for any $\pi_k \in \Pi_k$,

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

Define an *average* value function defined over X_i from h_i or *projection* of h_i with respect to the stationary distributions ρ_j over X_j of the selected policies π_j from the other agents $j \neq i$: for $x_i \in X_i$,

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

Similarly, define also an *average* reward function with respect to π_j: for $x_i \in X_i$, $a_i \in A_i$,

$$\bar{R}_i(x_i, a_i) := \sum_{x_{j \neq i} \in X_j} \left[\prod_{j \neq i} \rho_j(x_j)\right] \left(R\left(x, (\pi_1(x_1), \ldots, a_i, \ldots, \pi_N(x_N))\right)\right.$$

Since (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 for $x_i \in X_i$,

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

Denoting an action that achieves the right hand side of (1) as a_i^*, and summing up both sides over the stationary distributions of $\rho_j, j \neq i$, (1) can be rewritten as

$$g_i^* + \bar{h}_i(x_i) = \bar{R}_i(x_i, a_i^*) + \sum_{y_k \in X_k} P_i(y_i \mid x_i, a_i^*) \bar{h}_i(y_i).$$

It trivially follows that for $x_i \in X_i$,

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

Therefore, for $x_i \in X_i$,

$$g_i^* + \bar{h}_i(x_i) = \max_{a_i \in A_i} \left(\bar{R}_i(x_i, a_i) + \sum_{y_k \in X_k} P_i(y_i \mid x_i, a_i) \bar{h}_i(y_i) \right), \quad (5)$$

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 1, for all $x_i \in X_i$, there exists a constant κ and a bounded function ς defined over X_i for which

$$k + \varsigma(x_i) = \max_{a_i \in A_i} \left(R_i(x_i, a_i) + \sum_{y_k \in X_k} P_i(y_i \mid x_i, a_i) \varsigma(y_i) \right), \quad (6)$$

and if a constant κ' and a function ς' satisfy (6), then $\kappa' = \kappa$ (see Theorem 8.4.3 in [18]). Notice that g_i^* and $\bar{h}_i$ satisfy (6).

We summarize the localization concept below as a theorem:

Theorem 1: Given $\pi_j \in \Pi_j$ and corresponding $\{\rho_j(\cdot)\}$ for all agents $j \neq i$, under Assumption 1, for $M_i = (X_i, A_i, P_i, R_i)$ with $R_i = \bar{R}_i$, there exists a bounded function ς defined over X_i and a constant κ such that for $x_i \in X_i$,

$$k + \varsigma(x_i) = \max_{a_i \in A_i} \left(R_i(x_i, a_i) + \sum_{y_k \in X_k} P_i(y_i \mid x_i, a_i) \varsigma(y_i) \right), \quad (7)$$

and any policy $\pi_i \in \Pi_i$ which, for each local state $x_i \in X_i$, prescribes an action that maximizes the right hand side of (7) achieves g_i^* with $\kappa = g_i^*$.

We remark that if we add the condition that $\sum_{x_i \in X_i} \rho_i(x_i) \varsigma(x_i) = 0$ to (7), where ρ_i is the stationary distribution of the Markov chain induced from such a policy π_j for agent i as defined by the theorem, then $\varsigma = \bar{h}_i$ from the uniqueness of ς. As a special case of $\bar{R}_i$, if each agent k is associated with its local (bounded) reward function $R_k : X_k \times A_k \to \Re$ and $R(x,a) = \sum_k R_k(x_k, a_k)$, then $\bar{R}_i(x_i, a_i) = R_i(x_i, a_i) + \sum_{j \neq i, j=1,\ldots,N} \psi^{\pi_j}$, where ψ^{π_j} is the average reward of following the policy π_j with respect to $M_j = (X_j, A_j, P_j, R_j)$. Note that in case there is no constraint on the action choices across the agents, just solving the local MDP independently and forming a composite global policy provides an optimal joint policy for the global MDP. This particular case provides an intuitive argument for the results of the localization concept.

The localization theorem result is intuitively reasonable but also somewhat surprising. Suppose that an agent needs to maximize the global average reward constrained for a given set of fixed autonomous local policies of the other agents. If the joint state transition structure is factorial, maximizing the local average reward, defined with the projected reward function with respect to 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 the 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.

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

$$\psi^{\pi_i^n} + \varsigma^n(x_i)$$
$$= \bar{R}_i\left(x_i, \pi_i^n(x_i)\right) + \sum_{y_k \in X_k} P_i(y_i \mid x_i, \pi_i^n(x_i))\varsigma^n(y_i),$$

where this step is called *policy evaluation*. Then a policy π_i^{n+1} such that for $x_i \in X_i$,

$$\pi_i^{n+1}(x_i)$$
$$\in \arg\max_{a_i \in A_i}\left(\bar{R}_i(x_i, a_i) + \sum_{y_k \in X_k} P_i(y_i \mid x_i, a_i)\varsigma^n(y_i)\right)$$

is obtained, where this step is called *policy improvement*. Eventually, π_i^n converges to an optimal (in the sense of Theorem 1) π_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 give a pseudocode for it. Extending the algorithm into more than two agents is straightforward.

We first assume that the global joint reward function R is known to both agents. In other words, this information is given before the invocation of the algorithm. Each agent 1 and 2 starts with its own initial policy $\alpha^0 \in \Pi_1$ and $\beta^0 \in \Pi_2$. At the iteration $k \geq 1$, the agent 1(2) informs the agent 2(1) of $\alpha^{k-1}(\beta^{k-1})$ and the stationary distribution $\rho_1^{k-1}(\rho_2^{k-1})$ of $\alpha^{k-1}(\beta^{k-1})$. Then the agent 1 solves the local MDP $M_1^k = (X_1, A_1, P_1, R_1^k)$, where $R_1^k(x_1, a_1) = \sum_{x_2 \in X_2} \rho_2^{k-1}(x_1) R\left(x_1, x_2, a_1, \beta^{k-1}(x_2)\right)$. An optimal policy for M_1^k is α^k. Similarly, the agent 2 solves the local MDP $M_2^k = (X_2, A_2, P_2, R_2^k)$, where

$$R_2^k(x_2, a_2) =$$
$$\sum_{x_1 \in X_1} \rho_1^{k-1}(x_1) R\left(x_1, x_2, \alpha^{x-1}(x_1), a_2\right).$$

An optimal policy for M_2^k is β^k. Because at each iteration, each agent determines the best 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 underpinning 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 $m = 1, 2, \ldots$,

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

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 agents 1 and 2 monotonically improve in a zig-zag manner across the agents' views. We state this property as a proposition.

Proposition 1: For $k \geq 3$ and $k = 2m$ with $m = 1, 2, \ldots$, the following monotonicity holds: $g^{(\alpha_k, \beta_{k-1})} \geq g^{(\alpha_{k-2}, \beta_{k-3})}$ and $g^{(\alpha_{k-1}, \beta_k)} \geq g^{(\alpha_{k-3}, \beta_{k-2})}$.

Furthermore, because for $k \geq 1$,

$$g^{(\alpha_k, \beta_{k-1})} \geq g^{(\alpha, \beta_{k-1})} \quad \text{for any } \alpha \in \Theta_1$$

and for $k \geq 2$,

$$g^{(\alpha_{k-2}, \beta_{k-1})} \geq g^{(\alpha_{k-2}, \beta)} \quad \text{for any } \beta \in \Theta_2$$

the following holds:

Proposition 2: For $k \geq 2$ and $k = 2m$ with $m = 1, 2, \ldots$,

$$g^{(\alpha_k, \beta_{k-1})} \geq \begin{cases} g^{(\alpha, \beta_{k-1})} & \text{for any } \alpha \in \Theta_1 \\ g^{(\alpha_{k-2}, \beta)} & \text{for any } \beta \in \Theta_2 \\ g^{(\alpha, \beta_{k-3})} & \text{for any } \alpha \in \Theta_1 \\ g^{(\alpha_{k-4}, \beta)} & \text{for any } \beta \in \Theta_2 \\ \cdots & \\ g^{(\alpha_0, \beta)} & \text{for any } \beta \in \Theta_2 \end{cases}$$

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

$$g^{(\alpha_{k-1}, \beta_k)} \geq \begin{cases} g^{(\alpha_{k-1}, \beta)} & \text{for any } \beta \in \Theta_2 \\ g^{(\alpha, \beta_{k-2})} & \text{for any } \alpha \in \Theta_1 \\ g^{(\alpha_{k-3}, \beta)} & \text{for any } \beta \in \Theta_2 \\ g^{(\alpha, \beta_{k-4})} & \text{for any } \alpha \in \Theta_1 \\ \cdots & \\ g^{(\alpha, \beta_0)} & \text{for any } \beta \in \Theta_1 \end{cases}$$

It is then immediately true that at the best case, at the kth iteration with $k \geq 2$ and $k = 2m$, $m = 1, 2, \ldots$, we eliminate $O\left(k\left(|A_1|^{|X_1|} + |A_2|^{|X_2|}\right)\right)$ suboptimal policies from Θ.

At each iteration $k \geq 3$, the agent 1 checks the

condition that

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

is true or not and the agent 2 checks the condition that

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

is true or not. If both of the conditions are true, then both agents discontinue their own parts of the algorithm. Otherwise, both continue their communications and computations. This implicitly requires that each agent must inform the other agent of the flag of true or false, including the currently computed best reactive local policy and the stationary distribution of the policy. Then the question is if we ever have the instance where both of the conditions are true. Because of the monotonicity and the finite number of the (joint) policies in Θ, there exists an iteration that both agents acknowledge the true cases.

Even if both of the conditions are true, this does not mean that the performance of the converged policy sets of the agent 1 and the agent 2 are equal and either one of them is an optimal joint policy. This makes the algorithm converge to a local optimal policy and we need to select one of the converged policy sets of the agent 1 and the agent 2. Simply, we take the set having the better performance.

Suppose that $|A_i| = C$ and $|X_i| = D$ for all $i = 1,\ldots,N$. As we discussed before, when we apply the policy iteration algorithm to the global MDP, even if we ignore the autonomous policy structure condition, the time-complexity of solving the global MDP is $O(C^N \cdot D^{2N})$ for applying just one policy improvement step. The policy iteration algorithm converges to an optimal joint policy in $O(C^{N^{D^{2N}}})$ iterations at 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 to his local MDP, solving the local MDP takes $O(C^D D^2)$ at the worst case with the communication cost of $O(D)$. If our algorithm converged to an optimal joint policy for the global MDP at the worst case, it would have taken at most $O(C^D \cdot D^2 \cdot C^{N^{D^{2N}}})$ iterations. Therefore, the *worst-case* time complexity of solving the global MDP in a global manner can be alleviated by our algorithm depending on the size of C, D and N. Even though our algorithm converges to a local optimal solution in theory, the algorithm had better serve as a 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., [6, 7, 13, 14, 17 , 20]). Similarly, each agent can start with a good heuristic policy available for its local MDP or the given global MDP and apply one-iteration of our algorithm to generate an improved joint policy or apply k-iterations. Note that in contrast to one-step policy improvement, at the best case, each agent will eliminate the exponential number of suboptimal policies.

5. CONCLUSIONS

In this paper, we presented a novel concept of localization that can be used for breaking *the curse of dimensionality* when solving factorial MDPs in a multi-agents setting. A given global MDP is localized for each agent such that each agent is required to consider solving an MDP defined with only his local parameters. Based on the localization concept, we presented an iterative distributed algorithm for solving factorial MDPs. The overhead of the communication among agents and the complexity of solving each localized MDP in the algorithm will not be large if local state and action spaces are small. Furthermore, the result obtained by the algorithm provides a useful and meaningful global solution. The solution is an autonomous joint policy, which is a local optimal policy for the original MDP we want to solve.

The local optimality result of the proposed algorithm can be salvaged 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.

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

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

[10] O. Hernandez-Lerma, *Adaptive Markov Control Processes*. Springer-Verlag, 1989.

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

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

[13] 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.

[14] G. Koole and P. 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.

[15] 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.

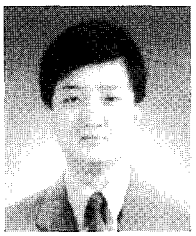
[16] M. Littman, T. Dean, and L. Kaelbling, "On the complexity of solving Markov decision problems," *Proc. 11th Annual Conf.* on Uncertainty in Artificial Intelligence, 1995, pp. 394-402.

[17] 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.

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

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

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



Hyeong Soo Chang received his B.S. and M.S. degrees in Electrical Engineering in 1994 and 1996, respectively and his Ph.D. degree in Computer Engineering in 2001, all from Purdue University, West Lafayette, Indiana. He was a Postdoctoral Research Fellow at the Institute for Systems Research at the University of Maryland, College Park from September 2001 to June 2002. He joined the Research Institute for Information and Communication Technology at Korea University in Seoul, Korea in June 2002 as a Research Professor. Since March 2003, he has been an Assistant Professor in the Department of Computer Science and Engineering at Sogang University in Seoul, Korea. His research interests include Markov decision processes, Markov games, controlled queueing processes, learning theory, and their applications to problems in communication networks, operations research, and artificial intelligence. He was awarded a graduate initiative research fellowship in 1997 from Purdue University.