Formalizing Multi-Agent POMDP's in the context of network routing

Bharaneedharan Rathnasabapathy and Piotr Gmytrasiewicz
Department of Computer Science
University of Illinois at Chicago
851 S. Morgan St, Chicago, IL
{brathnas,piotr}@cs.uic.edu

Abstract

This paper uses partially observable Markov decision processes (POMDP's) as a basic framework for Multi-Agent planning. We distinguish three perspectives: first one is that of an omniscient agent that has access to the global state of the system, second one is the perspective of an individual agent that has access only to its local state, and the third one is the perspective of an agent that models the states of information of the other agents. We detail how the first perspective differs from the other two due to the partial observability. POMDP's allow us to formally define the notion of optimal actions in each perspective, and to quantify the loss of performance due to partial observability, and possible gain in performance due to intelligent information exchange between the agents. As an example we consider the domain of agents in a distributed information network. There, agents have to decide how to route packets and how to share information with other agents. Though almost all routing protocols have been formulated based on detailed study of the functional parameters in the system, there has been no clear formal representation for optimality. We argue that the various routing protocols should fall out as different approximations to policies (optimal solutions) in such a framework. Our approach also proves critical and useful for the computation of error bounds due to approximations used in practical routing algorithms. Each routing protocol is a conditional plan that involves physical actions, which change the physical state of the system, and actions that explicitly exchange information.

1 Introduction

Decision making and planning using Markov decision processes (MDP's) is attracting attention in the research community for modeling many real-world decision problems. Researchers have used MDP's and POMDP's to formalize call admission control in cellular networks [5], management of ischemic heart disease [6] and robotics [2, 8]. POMDP's extend the MDP's and offer a very general formalism for representing real-world stochastic control problems, with explicit representation of partial observability of the environment and information sharing.

A concrete example of multi-agent environments to which POMDP framework is applicable is the domain of network routing. Networks can be viewed as a distributed system in which coordinated and informed decision making is crucial for optimal data routing. Network routing involves sending, receiving and forwarding data packets over a stationary or mobile network. The objective of the routing mechanism is to efficiently use the resources while performing the work of transporting data packets. Existing techniques attempt to capture the essential functional aspects of the system by the designer's pre-defining a set of rules through which information is exchanged and data is routed. As such, these approaches are subject to limitations of human intuition and insight. Having a formal model that fully represents the system and that defines the value of each course of action allows us to choose courses of action that are optimal in each state of the system.

We argue that optimal routing protocols should arise as solutions to POMDP's that represent the network routing problem. POMDP representation additionally allows us to compute error bounds due to approximating the optimal solutions to POMDP's [9]. The formal POMDP representation encompasses all the possible physical and communicative actions of the nodes in the network, the resources they have and the optimality criteria (or reward parameters in utility theory). This has a nice consequence that all complex information exchange and data exchange behaviors can be defined as a sequence of atomic actions that are defined within the POMDP.

Other formal approaches have been considered. Some use game theory [13], and other use utility-theoretic heuristics [10], MDP's [14, 5] and Q-learning [7]. Few, however, have approached the problem from the context of POMDP's and knowledge representation, and they fail to properly ac-

Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03)
0-7695-1874-5/03 $17.00 © 2002 IEEE

count for the benefits of actions that exchange information.

The rest of this paper concentrates on three different perspectives in which a distributed system can be viewed. First, we use MDP's to formalize the view of an omniscient agent, solution of which is objectively optimal. Second, we use POMDP's to represent the points of views of individual, locally rational agents, which have to make decisions based on their limited sensing capabilities. Such agents can perform physical actions and can gather information, but are incapable of efficient communication with other agents. We provide an initial experimental comparison of performance achieved by solving the above models of super-agent and locally rational agents. Finally, we extend the POMDP framework to model socially rational agents. These agents are capable of modeling the information available to other agents. The ability to model the other agent's knowledge allows the socially rational agents to effectively exchange information and to perform better than the locally rational agents. We expect that in the limit, if the communication is costless, the performance of the socially rational agents will reach the performance of the omniscient super-agent.

2 Markov decision processes (MDP's)

A MDP is a framework for representing a dynamic stochastic system. The system is dynamic because it can exist in any one of the several states at a particular point in time and it is stochastic because agents are uncertain of how their actions and external events change the state of the system. The Markov assumption on which MDP's are based states that the effects of actions and events on states depend only on the current system state. A Markov decision process is defined as a quintuple (S,A,E,T,R), where:

- S is the state space – a finite or infinite set of all physical states of the system.
- A is the set of actions that an agent can perform to effect the system state.
- E is the set of external events that the agent has no control over. These events can also change the state of a system.
- T quantifies the nondeterministic results actions and events have on the state of the system by specifying the probability of state transitions due to actions and events.
  T: $S \times A \times E \times S \longrightarrow [0,1]$
- R is the reward function describing the preference of certain states over others. It assigns real values to states.
  $R : S \longrightarrow \Re$

Viewing a whole distributed system as a fully observable Markov decision process corresponds to the perspective of the omniscient agent. We will go into the details of the omniscient agent in the next section after a small discussion about network routing.

3 Network Routing and Performance Metrics

A network usually consists of various nodes (computers, routers, etc.) interconnected by communication links. Network routing at the very basic deals with sending or forwarding data packets from source nodes to destination nodes. This seemingly simple task becomes more and more complicated with increase in network size and type. Data can be sent from one node to another through the links, which are not completely reliable. Each node has a limited buffer space used to temporarily store the data packets before forwarding them. If there is no space in a buffer, the incoming data packet is discarded.

This leads to the question of what features a routing protocol should control and optimize. In order to keep the model simple we consider data throughput and data delivery delay (packet delay) as the performance measure parameters. Several features defining the state of the network are relevant to the above performance parameters; they include network bandwidth, router buffer space, physical dynamics in network (mobility, power outages/switch offs) , incoming traffic volume, traffic density etc.

3.1 Routing as Markov Decision Processes : The Omniscient Agent

The omniscient agent observes the state information perfectly and dictates the course of action to all the nodes in the network. Modeling the routing problem as a MDP creates several issues. Our initial objective is to choose the variables that faithfully depict the system state and result in a model that is computationally solvable.[1]

3.2 The Physical State of the System

We assume that each data router or data routing entity has a buffer for storing the packets before transmitting them. The likelihood of a packet reaching the destination depends on the line quality and buffer space (if the buffer at the next hop node is full then the packet is dropped). Including buffer status as a part of the system state description is needed to capture this information. The line quality we

[1] These objectives are sometimes contradictory, and the point is to strike a balance between model size and complexity of finding solutions. Available computing power and sensitivity analysis are the helpful in finding the right trade-off.

mentioned is actually composed of various parameters. One of them is the distance between the nodes (in the case of a mobile network). In a mobile/radio network the distance between nodes directly influences the signal strength at receiver (transmission path loss) and in wired networks distance factor depends on signal attenuation. Thus distance is a measure derived from the location of corresponding nodes.

After the data is sent, the successful delivery of data needs to be recorded as part of the system state and reflected in the reward function, since the objective is to deliver packets to their destinations efficiently. Likewise, a packet dropped should also be penalized. All together, we use two variables for each node in the network; first variable represents the data packet reaching its destination at the node, and another variable registering that a data packet was dropped by the node.

We assume that the omniscient agent can direct nodes in the network to take two actions, either send a packet to a particular node or sit idle. If node 'a' is directed to send a packet it has for node 'c' through a node 'b', it would execute an action called Send $_{abc}$. Given other combinations of sources and destinations among N nodes, this results in (N-1)^2 actions for each node, and N $\times$ (N-1)^2 send actions and N idle actions in total.
Events also change the state of the system. These changes are stochastic in nature and agents have no control over them.

Transition probabilities describe the likelihood of a system transitioning from an initial state to another state after an action is performed or an event occurs. In the above Send $_{abc}$ action, the transition probability describes the likelihood of 'b' successfully receiving the packet from 'a' en-route to 'c'. Events can be modeled explicitly with separate transition probabilities or it can folded into that of actions.[2]

3.2.1 State space

State space S in the omniscient agent model is composed of the following variables.

- Location, L, of all nodes in the system: Describes the location of each node in the grid (mapped over the actual terrain, in case of a mobile network). A network having physically stationary nodes will have only one value of variable L.

  L_i = {1,2,...,C} describes the locations a node 'i' can be in, where C is the number of grid cells in the network. The total set of combinations of locations all nodes can be in is L = $L_1 \times L_2 \times L_3 \times$ ... L_N where N is the number of nodes in the network.

- B, Set of states of buffers of all the nodes: Each node has a buffer of size S_B. To describe packets in these buffers as well as the status of the buffers we associate a count of the number of packets outbound for each other node. The counts are restricted such that the total number of packets outbound in the buffer at a particular node equals the capacity of the buffer at that node.

  B_i is the set of permutations of { c_1, c_2, c_3, ... c_{i-1}, c_{i+1}, ... c_N } such that $\sum_{k=1}^{N} c_k = S_B$ and each $c_i \in \{0, \ldots S_B\}$. The total set of all buffer states in all nodes B = $B_1 \times B_2 \times B_3 \times$ ... B_N where N is the number of nodes in the network.

- Transmission Success Ts: This variable in a node denotes the successful delivery of a packet destined for a node. It is denoted by a variable T_{S_i} for a node 'i'. The maximum number of packets a node can receive at any time step is limited by the number of nodes in the network (leave alone the issue of having channel availability in a wireless medium). We can imagine each node maintaining a bucket into which packets meant for it are stored at each time step. This allows the node 'i' to measure the number of packets that were intended for and have reached node 'i'.

  T_{Si} = { 0, 1, .. N-1 }, total number of nodes in the network being N.

  The total set which includes all the nodes in the network $T_S = T_{S1} \times T_{S2} \times$ ... T_{SN} where N is the number of nodes in the network.

- Transmission failure Td: This variable in a node 'i' denotes the delivery failure of a packet destined for a node from node 'i'. This might be due to:

  - The node received a packet due to an event (ex. an application requesting to send some data across) but did not have any buffer space to hold the packet.
  - The node tried to send the packet to the next node in the chosen route to destination, but failed to send because of transmission errors that it immediately sensed (implicit observation in transition probabilities) during transmission.

  T_{Di} = { 0, 1, ... M }, the node can drop packets given to it by an event (ex. an application running on the node) or due to known transmission errors or both. If M_e is the maximum number of events that can occur in a time step and N-1 events can bring packets from other nodes then M = M_e + (N-1). The total set which includes all the nodes in the network $T_D = T_{D1} \times T_{D2}$

[2] The latter is known as an implicit-event model.

$\times$ ... T_{DN} where N is the number of nodes in the network.

The state of the system in its entirety depends on all the individual parameters representing the system's state. The state space is thus defined as, S={ L $\times$ B $\times$ T_S $\times$ T_D }

3.2.2 Actions

In the physical network that takes care of routing, the agent can be directed to perform two different action sets ; Send packets and be idle. Since we have not introduced variables capturing other resource states of the nodes, we ignore the idle action for the time being.

Let A_{Si} be the set of send actions i^{th} node can perform.

A_{Si} = N $\times$ N, such that for $a_{pq} \in A_{Si}$, p $\neq$ i $\neq$ q. The total number of actions each node can perform is $A_i = A_{Si}$

The total set of all actions (joint action set), A = { A_1 $\times$ A_2 $\times$ ... A_N } To simplify the structure in the resulting global MDP, we have initially assumed ordering of actions in the network. When one node performs an action in the network, other nodes perform a dummy idle action. Then we extend the same design to perform joint actions. The idea is to determine the globally optimal policy mapping states to a set of joint actions.

3.2.3 Events

In the omniscient agent model, we can have one event that changes the system state: The 'packet arrival' event modifies the state of the buffer or the packet drop variable (T_D) at a node. Each node can receive packets destined for every other node, this is due to applications on the nodes themselves generating requests (note that this is a global model, packets from other nodes arrive as a part of the send action). Therefore 'N' nodes can get 'N-1' types of requests each, leading to a total of N x (N -1) event types in the system. An arrival event is assumed to bring one data packet (one arrival with 'p' packets can be split as 'p' separate events). This event will update the buffer variable by incrementing the corresponding count, if there is space in the buffer. Otherwise the T_D variable for that node is set.

If E_i is the set of events that influences node i, then the total set of all events in the system E = { E_1 $\times$ E_2 $\times$ ... E_N }

3.2.4 Transition Probabilities

Transition probabilities describe the system dynamics resulting from actions and events. Send actions can be non-deterministic. A send action can fail if there is not enough buffer space in the next hop node. It may also fail due to transmission problems which are part of the environment over which the agents don't have control.

Without going into the details of how transition probabilities are computed, we will defined it as a function of $P_{transmission}$(s,s') and $P_{position}$(s,s'); the likelihood of the transmission leading to state s from s' and physical movement resulting in change of state to s.

$P_{transmission}$: Probability of packet reaching next hop node based on distance between transmitting and receiving node. This is based on power loss factor in transmitting medium (near linear relationship) and the availability of buffer space in the receiving node.

$P_{position}$: The locations of the nodes are a part of the system state. If the nodes are mobile then $P_{position}$(s,s') describes how likely each of the nodes will end up in the locations described by state s' given that they are in the locations described by state s. This is the mobility model describing the movement behavior of the nodes.

3.2.5 Rewards

Rewards are defined based on the values of the variables defining the state space. Every state with T_D (packet drop) variable set will incur a penalty and every state with T_S (delivery) set will be rewarded. At every stage, the total reward for the system would be
$\sum_{i=0}^{N}$ (T_{Si} - T_{Di}) , where T_{Si} denotes the transmission success variable at node i and T_{Di} denotes a drop.

3.2.6 Policy

The solution for an MDP maps the set of states of the system to actions. Policy π : S $\longrightarrow$ A associates an action with any given state. The omniscient agent uses the policy to dictate actions to all nodes in the network. The omniscient agent can use the policy since we defined it as being able to sense the state of the system perfectly.

3.2.7 Simulation

We used the problem solver written by Anthony Cassandra [3] for solving POMDP's but allowing complete observability to solve for MDP's. To measure the performance we calculate the average number of packets dropped in a 20-stage routing simulation over 10 trials with a constant traffic generator.

3.2.8 Environment

We modeled a 3-node network, each with buffer space enough to hold only one packet. The nodes can move inside an area divided into four cells and no two nodes at diagonally opposite ends can communicate. In order to keep the number of transition matrices small, the action space comprises of individual actions of each node (rather than a joint

[3] http://www.cs.brown.edu/research/ai/pomdp/code/ index.html

action set). At each time step one agent performs an action dictated by the policy and other two agents remain idle. A joint action can thus be broken down into 3-step action sequence.

3.2.9 Results

A segment of the computed policy is shown below

$1.State : (0,1,0)(10,00,00)([X,1,0][0,X,1][0,0,X])$
$Action : \{1,3,2\}$
$2.State : (0,1,1)(00,00,00)([X,0,0][0,X,1][0,0,X])$
$Action : \{2,3,3\}$
$3.State : (1,2,3)(00,00,00)([X,0,0][0,X,0][0,1,X])$
$Action : \{3,2,2\}$
$4.State : (1,2,3)(00,00,00)([X,0,1][0,X,0][1,0,X])$
$Action : \{3,2,1\}$

The 'X' in the buffer state above denotes that each node cannot buffer a packet meant for itself. Each state-action pair above describes a course of action taken at a given state. The state description is a structure composed of

(location of node1, node2, node3)
(Ts Td flags in node1,node2,node3)
(Buffer states in node1,node2,node3)

The action picks the node to perform an action and the type of action. {Node performing the action, Send to node, Destination node}

We can see that if a node is close by then the packet travels in one hop as in rule 2. But if the destination is queued and full (only one space is available at each node), then the packet is juggled and sent to destination via another node, as in rule 1. Intuitively we can see that this is the ideal way to perform the tasks. The solution to this network with joint actions would, in fact, result in optimal routing performance under our design (since it is the optimal solution to the MDP).

The solution describes how an omniscient agent would work to achieve optimum results. Such an all-observing oracle would direct all the nodes to preform actions optimal in each physical state. Of course reality is different. In the next few sections we will explore how to model uncertainty in a node's perspective about the system state, giving locally rational agents.

4 Locally Rational Agents Model

Assuming each node knows the underlying physical model (such as number of nodes in the network, their buffer space etc), we can model each node's view of the system as its belief about (or probability distribution over) the possible states of the system at a given time.

The set of 'belief states' an agent can be in is the set of all probability distributions over S, Δ(S). A belief state s $\in$ Δ(S) is in-fact a point in the |S | dimensional probability space.

4.1 Partially observable Markov decision processes (POMDP's)

There are a few differences between the MDP and POMDP formalisms. POMDP's offer more general view than MDP's that helps us model the uncertain nature of sensing actions of the agents. We introduce two new parameters, observation set and observation probability. POMDP captures uncertainty in both actions and the state (the agent now does not know what the system state exactly is). The POMDP is formally defined as the tuple $< S, A, E, T, R, \theta, O >$ where

- T defines the transition probabilities or the uncertainty associated with the outcomes when actions are taken or events happen.
  T: S $\times$ A $\times$ E $\times$ S $\longrightarrow$ [0,1]
- θ is the finite set of observations an agent(or node) can make.
- O defines the observation probabilities that models the likelihood of making an observation when the system is in a given state, O : $\theta \times S \longrightarrow [0,1]$.

  The solution to the POMDP maps the set of all probability distributions over the state space to the set of agent's possible actions.

  Policy π : Δ(S) $\longrightarrow$ A_i

4.2 Observations and Actions in the network

Nodes in network can make observations that give information about the environment. They can perfectly observe their buffer state, their current location, if they have dropped a data packet, etc. In addition to this, the nodes can observe the actions of other agents using which they can extract the non-local system state information with some uncertainty. Every other node's actions are external, hence they are modeled as events. These events provide information though not explicitly. For example in the 3-node network with a single buffer space, if a node 'b' sends a data packet to node 'a' then node 'a' knows with some uncertainty that node 'b' has a free buffer space. These are the kinds of actions that have implicit information content. Each node has one unique observation for each of its own physical state and one for each type of event. If O_s is the set of observations due to the agents own state (perfectly observable) and O_E be the set of observations due to all external events then $\theta = O_s \times O_E$. The set of actions in each of the N POMDP models (for each node) is the set of actions each agent can perform.

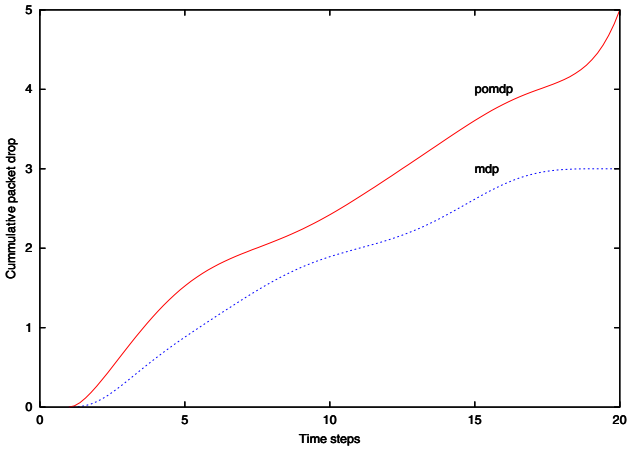

Figure 1. Comparison of MDP and POMDP based routing

4.3 Discussion of the solution to the POMDP

We used the same network topology with a symmetric model (solving for one would solve for all the 3 agents) that is partially observable.

We will not go into details of how the POMDP's are solved as it would constitute a detailed discussion by itself [6, 4]. The POMDP here describes the network model of each node having knowledge about the underlying physical structure and its own physical conditions but having imperfect information about other nodes. Solving this model for every node in the network, we would have an action prescribed by the policy computed for every system state or exactly belief state (or state of information of each node). Given that the nodes start with arbitrary belief states, new belief states are computed based on the observation they receive, the actions they perform and the current belief state. The 'value function' resulting from the POMDP model would dictate the actions chosen at a given belief state. The value function describes the expected value of choosing an action given the current belief state. The solution actually defines the value of a set of conditional plans, each starting with some action. These conditional plans dictate the choice of action given the current observation. Why have many plans in the solution if we are talking about the optimal plan ? Well, each of these conditional plans is optimal at some particular information state of the node, i.e. the information state at which the agent starts using the plan.

Looking at the simulation results in Figure 1., we see that the POMDP based locally rational agent model without explicit information exchange starts dropping more and more packets as time goes on. This is expected as agents cannot effectively exchange information about other nodes. This shows the need to capture more information content at each agent level for efficient routing.

5 Multi-Agent POMDP : Socially Rational Agents

Let us consider a network that has three nodes a,b and c. If node 'a' did not know where node 'c' is or how to reach 'c', it could ask the nodes adjacent to it(node b). In the worst case if it didn't know about any other node it can broadcast a query (similar to what humans do, asking people you know about the information you are seeking. If you are a complete stranger then post your question on a public location,like a bulletin board etc.). Whoever receives the query and knows the answer to the question will find it beneficial to reply.

If nobody responds then it gives us two kinds of information; either the node that broadcast the query was isolated from other nodes or all nodes that know the information are isolated, as in the case of network partitioning. There could also be several other reasons, such as the query data being lost during transmission but we will limit the possibilities to keep the model simple for discussion. If the node does receive the information it has asked for it can update its belief about the network and proceed with performing the most rewarding action.

To choose the most rewarding plan, each agent would need to know how other agents would react to their own beliefs about the system state. This model assumes that all agents have the same utility function and each of them knows the observation function of the other agents in the system. In this discussion we limit the model to actions that implicitly exchange information in the same time interval and those that exchange information in one time-step alone. Information query actions involving multiple time-steps opens the issue of observation delays which is a complication best not handled here. Given this scenario, we will look at the formal representation of the multi-agent POMDP.

5.1 Definition of Multi-agent POMDP's

This model is similar to the one in previous subsection. Each agent has a POMDP model that is described by the tuple M_i=$< S_i, A_i, E_i, T_i, \theta_i, O_i, R_i, M_{-i} >$.

Let Δ(S) be the set of all probability distributions over S, the set of physical states. We define $\Delta(S \times_{N-1} \Delta(S))$ as the set of states defining an agents belief's about the physical states and the beliefs of other agents about the physical states. Along the same lines, $S_i = S \times_{N-1} \Delta(S) \times_{N-1} \Delta(S \times_{N-1} \Delta(s))$, where $\Delta(S_i)$ is the set of states of an agent's beliefs of the systems states, beliefs of other agents and their beliefs about it. This enables the agent 'i' to monitor its beliefs based on the physical state of the system and

the belief states of other agents in the system. [1] discusses knowledge hierarchies in detail. The belief hierarchies extend to infinite depth with decreasing information content at each level. For reasons obvious, we restrict ourselves to second-level beliefs that capture an agents belief about other agent's belief about its physical state.

A_i = The set of all actions that agent 'i' can perform. These actions could be physical actions that change the physical state of the system (and could also the change information state of other agents). We include an action NULL in which case the agent remains idle.

$E_i = \{\times A\_i\}$ is the set of all actions that can be performed by agents other than 'i'.

$T_i : S_i \times A_i \times E_i \times S_i \longrightarrow [0, 1]$, describes how each action and events changes the state (with respect to the agent 'i'). This is the crucial part of the model, given that each agent knows the observation functions of other agents and their utility functions. The computed policy for every other agent gives the agent information about the likely actions chosen by other agents in different belief states.

θ_i is the set of observations perceived by agent 'i'.

$O_i : S_i \times \theta_i \times A_i \times E_i \longrightarrow [0, 1]$ describes how likely each agent 'i' can perceive observations at a given state and action or event. Note that observations can be a result of external events in a more general framework. For example touch is the only way of perceiving, then the agent can feel if it touches something or something touches him. A more simpler way of defining this would be to enrich the state space, which we will avoid doing here.

$R_i : S_i \longrightarrow \Re$, is the utility function of agent 'i' describing the value of each state.

$M_i = \{M_{j \neq i}\}$is the set of models (the states, observation functions, observations, utility functions, actions, events and transition probabilities of all agents other than 'i') of other agents.

Since the system state depends on the effect of the joint actions of all the agents, each agent has to know what every other agent will most likely do. Intuitively this can be phrased as "A believes that B,C,D,... believe something about the system and every other agent. Based on A's beliefs, A knows that B,C,D,... would perform actions with some probability each. This allows A to choose an action such that the resulting joint action (from A's perspective) has the highest expected utility".

The policy in this model is a mapping from the set of probability distributions over S_i to the set of actions A_i, $\pi : \Delta(S) \longrightarrow A_i$.

5.2 Solution approximation over finite state space

Computing optimal policies for POMDP's has been the bottleneck in modeling real-world decision problems using the framework. [11] show that for a finite horizon decision problem in the framework is PSPACE-hard. For detailed discussion on complexity issues we refer the reader to [3] and [11]. However polynomial time approximation algorithms exist, one of which is the grid based approximation. Refer to [9] and [15] for discussion about grid based techniques which can approximate solutions to the problem for finite S_i. Instead of computing the value for the whole belief state as in exact algorithms, we compute the values of a set of belief states in the belief space and approximate the values of the other belief states. This is done by determining a convex combination of belief points that surround the belief state of interest and using their values to compute the value of the belief state. If the set of belief states were to be visualized as a N-dimensional grid in N-dimensional space, all we have to do is to determine the smallest sub-simplex that contains the belief point for which we have to compute the value. The problem is therefore reduced to computing the values of those selected set of belief points using the regular gauss-seidel value iteration algorithm.

Additionally, [9] shows that the error bounds for the value function can be found by computing the lower bound and upper bound of the value function. Once we know the upper and lower bounds, the difference gives us an idea of how tight the error bound is.

5.3 Conclusions and future work

In this paper we have presented a preliminary framework to formalize network routing as a multi-agent decision problem using POMDP's. We include all possible atomic physical actions that agents/nodes can perform and describe how they influence the state of the system thereby allowing a functional protocol (rules suggesting courses of actions based on system state) to fall out as an optimal policy of the POMDP. The ability to determine the error of our approximation in POMDP's helps us in computing near-optimal solutions. A detailed study of solution techniques for POMDP's show us the tractability issues in huge models. Grid-based approximation schemes are again limited by memory complexity introduced by the state space. Computation of error resulting from approximation of both state space and belief space in POMDP's needs to be studied. Promising directions for future work are to use interpolation algorithms in infinite dimensional space for approximating value functions of POMDP's and using finite histories(with known priors) instead of handling the entire belief space.

In a related research work, [12] have recently developed a communicative multi-agent team decision problem framework that is very similar to multi-agent POMDP. A COM-MTDT from a multi-agent POMDP perspective is a model with single level of knowledge hierarchy. A model in which agents only maintain the physical state space information, not information about other agents state of knowl-

edge and messages (actions that explicitly exchange information about the states) only update each agent's belief about the physical state space. An empirical comparison between multi-agent POMDP's and COM-MTDP's should give interesting insights into both approaches. This will be pursued in our future work.

References

[1] R. J. Aumann. Interactive epistemology : Knowledge. In *International Journal of Game Theory*, volume 28, pages 263–300, (1999).

[2] C. Boutilier, T. Dean, and S. Hanks. Decision theoretic planning: Structural assumptions and computational leverage. In *Journal of Artificial Intelligence Research*, pages 11:1–94, November 1999.

[3] D. Burago, M. de Rougemont, and A. Slissenko. On the complexity of partially observed markov decision processes. In *Theoretical Computer Science, 157(2)*, pages 161–183, 1996.

[4] A. R. Cassandra. Exact and approximate algorithms for partially observable markov decision processes. In *Ph.D. Thesis. Brown University, Department of Computer Science, Providence, RI*, 1998.

[5] Z. Haas, J. Y. Halpern, L. Li, and S. B. Wicker. A decision-theoretic approach to resource allocation in wireless multimedia networks. In *Proceedings of Dial M for Mobility*, pages 86–95, 2000.

[6] M. Hauskrecht. Planning and control in stochastic domains with imperfect information. phd dissertation. In *MIT-LCS-TR-738*, 1997.

[7] S. Kumar and R. Miikkulainen. Confidence based dual reinforcement q-routing: An adaptive online network routing algorithm. In *International Joint Conference on Artificial Intelligence, Stockholm, Sweden.*, July 1999.

[8] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling. Learning policies for partially observable environments: Scaling up. In *Proceedings of the Twelfth International Conference on Machine Learning*, 1995.

[9] W. S. Lovejoy. Computationally feasible bounds for partially observed markov decision processes. In *Operations research*, volume 39, No. 1, 1991.

[10] A. R. Mikler, V. Honavar, and J. S. Wong. Utility-theoretic heuristics for intelligent adaptive routing in large communication networks. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1996.

[11] C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of markov decision processes. In *Mathematics of Operations Research, 12:3*, pages 441–450, 1987.

[12] D. V. Pynadath and M. Tambe. Multiagent teamwork: Analyzing the optimality and complexity of key theories and models. In *Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2002.

[13] K. Srikanth. Network routing problems using game theory. In *Technical report , Indian Institute of Technology.*, April 2001.

[14] G. Wu, E. K. Chong, and R. Givan. Congestion control via online sampling. In *Proceedings of INFOCOM*, 2001.

[15] R. Zhou and E. A. Hansen. An improved grid-based approximation algorithm for pomdps. In *Proceedings of International Joint Conference in Artificial Intelligence*, 2001.