Computational Models for Decision Making in Dynamic and Uncertain Domains

Omid Madani

Abstract

Decision making in dynamic, uncertain environments has traditionally been a rich source of many important mathematical and computational problems. In this report, I will describe three models of control in dynamic systems consisting of a finite set of states where decisions influence state transitions. I will explain control objectives and motivations for the models, and will discuss open problems on computational complexity and algorithmic issues and possibilities for hybrid models.

1 Introduction

Problems in control and decision making in a dynamic environment have been a source of rich theoretical and practical research in several disciplines [NM80, DW91]. A good portion of this work has traditionally focused on understanding models and model consequences. Recently many researchers, interested in implementing systems based on such models, are paying more attention to related computational tractability issues. The growing power of our information processing systems is providing ample opportunity and motivation to refine and extend the old models and explore new ones in order to achieve more sophistication in system behavior.

Dynamic decision making models and related problems have had an impressive record in stimulating fruitful research, as the following quote by George Dantzig, one of the pioneers of the widely applicable linear programming model, suggests in reference to the model [Dan91]: "... it is interesting to note that the original problem that started my research is still outstanding — namely the problem of planning or scheduling dynamically over time, particularly planning dynamically under uncertainty. If such a problem could be successfully solved it could eventually through better planning contribute to the well-being and stability of the world." At this point, the field is rich with a variety of models that have their own characteristics and application areas.

I will describe a few such models in this report. The models share several characteristics that are roughly as follows. Control is exerted in a *system* given as part of the problem statement. The system is characterized by a set of states and transitions between them. In this report, we will always assume that the set of states is finite[1]. At each point in time the system is in exactly one state and the system may make transitions from one state to another at discrete time intervals. We assume that the system characteristics are known. Each model includes one or more decision makers, sometimes referred to as controllers or players when appropriate, who can exert control by affecting system transitions. In several of the models the transitions can solely be caused by the controller and in one model some transitions can only be prevented. The objectives of the control problems are numerous and can range from reducing the cost of the system operations, guiding the system towards a designated target state, or ensuring desirable properties such as safety.

One popular genre of models with a rich mathematical theory and broad applicability is the classic *Markov Decision Process* (MDP) models. The concepts behind these models date as far back as the calculus of variations problems of the 17th century. Many researchers including Masse, Walde, Bellman, Shapley and Karlin conducted the research that originated the field in the forties and fifties (see [Put94] for a historical background). Applications of MDPs are diverse: MDPs are used to model problems in inventory management, highway pavement maintenance, quality control and more (see for example [Put94] and [Whi88]). The partially observable MDP (POMDP) is an important generalization where the decision maker has only partial information on the current state of the system. Applications of such models are again diverse and include medical decision making, design of teaching systems, cost control in accounting and equipment maintenance and replacement [Put94, Mon82]. Recently, computer scientists have looked at these models as a framework for planning under uncertainty [BDH95] and the models find applications in areas such as robot navigation [BRS96, KGS95] and medical decision making [Hau97].

There are several modeling questions and open algorithmic problems about MDPs and their generalizations. The computational complexity of solving MDP problems was addressed only relatively recently beginning with the work in [PT87]. POMDPs have been shown to be intractable to solve in general [PT87, Lit96], which partially explains the limited use of these models in practice so far. However we report on work in [BRS96] showing that studying restricted models can be fruitful for designing approximation algorithms. There remain questions on different restrictions on partial observability and whether these restricted models and perhaps their combinations are effective in modeling real world problems. In [Con92], Condon investigates a two player game generalization of the MDP model. The research brings out several outstanding open problems on the existence of polynomial time algorithms not only for the game model but also for MDPs. Further work in this area may lead to novel and perhaps more efficient algorithms for MDPs and their many variants in addition to settling several theoretical questions.

We will also look at a model of control called the discrete event system (DES) started and mainly developed by Wonham and his students [RW87]. While in an MDP problem, the source

[1]However, many variations of the models have been investigated for which the state space is infinite.

of change in state is a controller, in a DES it is mostly the "outside" world independent of the controller (*e.g.* the users of the system), and the controller takes the role of the supervisor of these changes called events. The supervisor might intervene and prevent some of the changes under its control. DES models have found practical control applications in areas such as protocol design for communications network and database operations. My treatment of the DES formalism is brief. The hope is to understand enough of the model to be able to contrast the MDP and DES models in their choice of objective criteria and solution characteristics.

1.1 Paper Overview

In Section 2, I will introduce several MDP model concepts as I report on the work in [BRS96]. In Section 3 I describe the game generalization of MDPs. Each of these sections concludes with a discussion of related work and interesting open problems in the area. I present some basics of DES systems in section 4 and will go over a variation called a discrete event stochastic system (DESS) in 4.3. We will find out that the version of DESS treated in 4.3 can also be thought of as a variation on an MDP problem. I will close with a discussion on relating the models in Section 5 and conclude in Section 6.

2 MDPs and POMDPs

In this section, by going through the work in [BRS96] as an example, I will familiarize the reader with important concepts in Markov decision process models (MDPs) and their generalization, partially observable Markov decision processes (POMDPs). As sufficient background is built, I will describe the problems that [BRS96] and others have addressed.

The problem, which I will subsequently refer to as the *target* problem, begins with the following model description: given is a tuple $S = (Q, C, clr, \Sigma, T, s, t)$ where Q is a set of n states, C is a set of colors, $|C| \leq n$, $clr : Q \rightarrow C$ is an onto (coloring) function, Σ is a finite set of actions, $T = \{T_a | a \in \Sigma\}$ is a set of stochastic $n \times n$ transition matrices T_a, one for each action $a \in \Sigma$, and s and t are start state and the target state respectively ($s, t \in Q$). The objective of the target problem is to come up with a strategy that maximizes the probability of reaching the target state from the start state in no more than an specified K number of action execution steps.

Here is a description of how a problem of such nature may arise that will in addition make the semantics of the transition matrices and the color set clear. Imagine that each state represents a location of a robot, and the robot wants to go from the start location s to the target location t. Note that the possible set of locations is modeled (perhaps approximated) by a finite set. The actions in Σ comprise the set of actions available to the robot, and the uncertainty of the effects of these actions is reflected in the stochasticity of the transition matrices in the set T. There is an additional source of uncertainty. The robot may not know its exact location, *i.e.*

two different locations can give it the same color (sometimes called an observable or signal). Each location (state) has a unique color, but each color may correspond to several locations. The coloring function *clr* determines the mapping of locations to color.

If the robot is in state i and executes action a, then with probability $T_a[i,j]$, it will end up in state j, where $T_a[i,j]$ is the entry in matrix T_a appearing in the row i corresponding to state i and in column j corresponding to state j. The fact that the next state is only a probabilistic function of just the current state and the action executed in the current state (and, for instance, not on how the current state was arrived at) is the so-called "Markov property", and the name Markov decision process coined by Bellman (see [Put94]) originates from this property. We are assuming that each state is observed only indirectly, through the color it emits (partial observability). Hence to find out the probability that it is in state j, the robot should make a Bayesian update depending on the color it sees: $p(j|i,a,c) = \frac{p(j|i,a)}{\sum_{q \in Q, clr(q)=c} p(q|i,a)}$, where $p(j|i,a,c)$ denotes the conditional probability that robot is in state j, given its previous location (state) was i, it executed action a, and it is observing color c, and $p(j|i,a)= T_a[i,j]$. More generally, the robot may have a distribution over its current set of states, represented by a vector of probabilities $\vec{p}$, where $\vec{p}_i$ is the probability that current state of the system is i. Then $p(j|\vec{p},a,c) = \sum_{i \in Q} \vec{p}_i p(j|i,a,c)$.

I briefly address a few aspects of the model, including what we mean by a strategy for the robot, before discussing the computational complexity and algorithms.

2.1 Observability

Two extreme cases of partial observability are when the system is fully observable, that is the color function is bijective ($|C| = n$), and when the system is *unobservable*, when all states emit the same color ($|C| = 1 < n$). In the classical MDP problem, the assumption has traditionally been full observability, and I will refer to a fully observable model as an MDP, to distinguish it from the more general partially observable model, the POMDP.

In the literature it is usually the case that the color is a probabilistic function of state (see subsection 2.6 and Figure 2 for an example), however such a problem can be converted to a POMDP with a deterministic color function where the new states are designate by (state, color) pairs of the original problem: (i,c) has color c. Hence the new state space will have cardinality at most $n|C|$.

2.2 Finite vs. Infinite Horizon

For the above problem, we are asking for a strategy that maximizes the probability of reaching the target state in no more than K (generally a polynomial in n) steps. This is called a *finite horizon criterion*, and it is the problem treated in [BRS96]. Alternatively, we may ask for a strategy that simply maximizes the probability of reaching the target, regardless of the number of steps. This is an example of a problem with the *infinite horizon criterion.* As we shall see, the open problem in Section 3 is an infinite horizon criterion question. Choice of the horizon criterion depends on the nature of model and perhaps other considerations such as tractability.

2.3 Types of Strategies

An important question in any control problem is the choice of the class of strategies to consider, that is, on which aspects of the system should the decision maker base its decisions to ensure optimal or near optimal behavior. Naturally, we want the class to be as restricted as possible for the purposes of simplicity and tractability of computation. The type of strategies considered depends on characteristics of the problem, optimality criteria, and in some cases on the computational constraints.

Let us consider the fully observable (*i.e.* the MDP) case (when *clr* is bijective) in the above model. It is not hard to see that in this case optimal strategies need be dependent on the state of the system and the number of steps to completion (number of action executions remaining). These types of strategies are called *state-based* or *positional* strategies. Figure 1a illustrates a positional strategy.

Let $v_k(i)$ denote the maximum probability of reaching the target state in k or fewer steps starting in state i. We refer to $v_k(i)$ as value of state i for k steps. These quantities are by definition state dependent only and a dynamic programming algorithm for computing these values readily suggests itself. We have $v_0(t) = 1$, and $v_0(i) = 0$ for $i \neq t$, and the algorithm uses the update formula:

$$\forall i \in Q, v_k(i) = \max_{a \in \Sigma} \Sigma_{j \in Q} p(j|i,a) v_{k-1}(j) \tag{1}$$

An action is optimal for state i, with k steps to completion, if it maximizes the right hand side of equation 1.

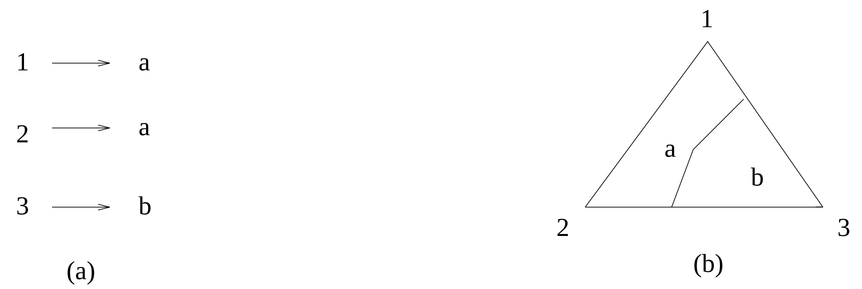

Figure 1: (a) A positional strategy for a 3 state MDP. At states 1 and 2 action a is prescribed, and at state 3 action b. (b) A partition of a 3 dimensional unit simplex as a strategy for a 3 state $POMDP$. Action b is prescribed at distributions with a high probability for state 3.

Now let us consider the case of partial observability ($m < n$). In this case, to act optimally, it is not enough to know the best action at each state since as we saw the decision maker may only have a distribution over states during the course of action executions. An inductive argument similar to above shows that, for a given number of steps to completion, a mapping from the unit simplex of distributions over states to optimal actions and values (mapping from probability vectors to actions) suffices. Though there are infinitely many distributions over actions (infinitely many points in the simplex), it is not hard to show that the number of connected regions where the same action is optimal is finite in finite horizon problems (see figure 1b) and the borders between these regions are linear [SS73]. Figure 1b shows a 3 dimensional unit simplex representing the set of probability distributions over 3 states. The simplex is partitioned into two regions with a piecewise linear border. A dynamic programming algorithm can compute optimal partitions but unfortunately it may run in exponential time. In fact, this inefficiency is likely to be an inherent property of the problem: POMDPs are hard to solve in general (see below). The approximation algorithm in [BRS96] is based on an approximation of the optimal partition of the simplex.

2.4 Computational Complexity

Substantial work on MDPs and POMDPs began in the 40's and the 60's respectively. However, the computational complexity of the problems had not been addressed until recently. Even today it is not known whether the widely used strategy improvement algorithm for (infinite horizon) MDPs runs in polynomial time. Lovejoy [Lov91] surveying computational aspects of POMDPs notes, "The significant applied potential for such processes remain largely unrealized, due to an historical lack of tractable solution methodologies."

The study of computational complexity of MDPs and POMDPs began with the work in [PT87] who show that many variants of the classical fully observable MDP problems, both for the finite and the infinite horizon, can be solved in polynomial time using dynamic programming or linear programming methods, and in fact the problems are P-complete, where the input length consists of the number of states and actions and the encoded representation of the entries in matrices. Unfortunately, in the case of partial observability, the problem becomes very hard. In [PT87], it is shown that the problem of coming up with an optimal strategy for partially observed problem is $PSPACE$-hard even when the finite horizon is restricted to be n. Furthermore, even if an optimal such strategy is constructed (perhaps off-line given enough time), unless $PSPACE = \Sigma_2^p$, it is impossible to get a strategy of length polynomial in the description of the MDP process, such that the decision maker, given the observation symbols and the strategy, can compute what to do next in polynomial time.

With unobservability the problem seems to be slightly easier than partial observability: in [PT87] it is shown that the unobservable problem with a horizon of n is NP-complete. Still, perhaps surprisingly, even a weak approximation is hard, as is established in the following theorem appearing in [BRS96], proved by using a reduction from the $3SAT$ problem. Notice that in the unobservable case, we need only ask whether a sequence (string) of actions exists that has sufficient probability of taking the system from the start state to the target.

Theorem 2.1 *The following problem is NP-hard: Given is an unobservable POMDP, with n states, a starting state s and a target state t such that one of the following two cases is true:*

1. *There is a string of actions of length n or less that takes the system from state s to t with probability 1, or*
2. *the probability is less than $exp(-\sqrt(n))$ for any such string.*

Decide whether case 1 holds.

Given the hardness of these problems, one way to approach the problem of partial observability is to come up with plausible restrictions so that computing optimal or approximate strategies become tractable. In the next section, I'll go over one such model investigated in [BRS96].

2.5 A Restricted Problem and an Approximation Algorithm

The hardness results above show that unless we restrict the uncertainty in the unobservability of the system, partially observable problems will very likely remain intractable. One plausible assumption on a partially observable system is the following: during the course of execution, the decision-maker may not know the exact system state, but the possible system choices (that have high likelihood) may be few. For example, the decision-maker may always be able to narrow down the choice of current system state to only 3 states. For such cases the following restricted POMDP model is appropriate:

Assume that each color represents at most a constant m number of states: for any color c, let $Q_c = \{q \in Q | clr(q) = c\}$, and we assume for any color c, $|Q_c| \leq m$, for some constant m independent of n. The number m is called the *coloring multiplicity.*

Let us consider the target problem with the above restriction. The complexity of exactly solving the problem remains NP-hard for $m \geq 3$, shown by a nontrivial reduction from the partition problem[2] [BRS96]:

Theorem 2.2 *Constructing an optimal strategy for the target problem with coloring multiplicity 3 is NP-hard.*

However partial observability with constant multiplicity admits a polynomial time approximation algorithm in the following sense: the strategy σ output by the algorithm is $\epsilon-$optimal, *i.e.* the probability of reaching the target using σ in a specified number of steps is at most ϵ less than the probability of using the optimal strategy. The approximation algorithm runs in time polynomial in n, $1/\epsilon$ and horizon K. I will next describe the algorithm in [BRS96].

Without loss of generality, assume each color has multiplicity exactly m. Enumerate the states in Q by two indices, the first index being a color, so that $(c, i) \in Q_c$ denotes the ith state

[2]The case of $m = 2$ is open.

with color c, where $1 \leq c \leq |C|$, and $1 \leq i \leq m$. The set of distribution vectors over each Q_c forms an m dimensional unit simplex denoted by X here. For a color c, by $\vec{x}/Q_c$, I mean the probability distribution $\vec{x} \in X$ over states of Q_c, in which case $\vec{x}_i$ is the probability of being in state (c, i). Let $v_{k,c}(\vec{x})$ be the maximum probability of reaching target t starting with $\vec{x}/Q_c$ in no more than k steps. Let $\|.\|$ denote the l^1 metric for X: $\|\vec{x}\| = \Sigma|\vec{x}_i|$, for $\vec{x} \in X$. Call a real valued function over X *Lipschitz*-1 if $|f(\vec{x}) - f(\vec{y})| \leq \|\vec{x} - \vec{y}\|$, for any $\vec{x}, \vec{y} \in X$. The following result follows from the intuitive fact that changing the probability of any state by some amount ϵ can change the maximum proability of reaching the target by at most ϵ.

Lemma 2.3 *[BRS96] All $v_{k,c}$ are Lipschitz-1 functions.*

The functions $v_{k,c}$ satisfy the following recurrent system of equations similar to equation 1:

$$v_{k,c}(\vec{x}) = \max_{a \in \Sigma} \Sigma_{c'=1}^{m} p(c'|\vec{x}/Q_c, a) v_{k-1,c'}(T_{c,c'}(\vec{x}, a)),$$

where $p(c'|\vec{x}/Q_c, a)$ denotes the probability that color c' is observed given that action a is executed with a prior $\vec{x}/Q_c$, and $T_{c,c'}(\vec{x}, a)$ is the conditional distribution over states in $Q_{c'}$ given that c' is observed. They are computed as follows:

$$p(c'|\vec{x}/Q_c, a) = \sum_{i=1}^{m} \sum_{j=1}^{m} \vec{x}_i p((c', j)|(c, i), a)$$

and

$$(T_{c,c'}(\vec{x}, a))_j = \frac{\sum_{i=1}^{m} \vec{x}_i p((c', j)|(c, i), a)}{p(c'|\vec{x}/Q_c, a)}$$

Now let $\delta = \frac{\epsilon}{2K}$, where K is the specified number of steps and ϵ is the chosen approximation precision. Let M be the smallest integer greater than $1/\delta$. The algorithm subdivides X into M^{m-1} equal simplexes by hyperplanes parallel to the faces of X. Consider the class V of continuous real-valued functions on X whose restriction on every tiny simplex of the X partition is linear. For a real-valued function f on X, denote by $sample(f)$ the unique function from V that coincides with f on all the vertices of the simplexes of the partition. Let $\|.\|$ be the supremum norm on real valued functions on X : $\|f\| = sup_{\vec{x} \in X}|f(\vec{x})|$. We note that if f is a *Lipschitz*-1 function then $\|f - sample(f)\| \leq \delta$.

The algorithm: Define the value functions $\tilde{f}_{k,c} : X \rightarrow R, 0 \leq k \leq K$, and the decision functions $d_{k,c} : X \rightarrow \Sigma$, $k \geq 1$, as follows. $\tilde{f}_{0,c}(\vec{x}) = 0$ if target $t \notin Q_c$, otherwise if t is denoted by (c, j) in Q_c, then $\tilde{f}_{0,c}(\vec{x}) = \vec{x}_j$. For $k > 0$, let

$$\hat{f}_{k,c}(\vec{x}) = \max_{a \in \Sigma} \sum_{c'=1}^{m} p(c'|\vec{x}/Q_c, a) \tilde{f}_{k-1,c}(T_{c,c'}(\vec{x}, a)), \tag{2}$$

and let $\tilde{f}_{k,c} = sample(\hat{f}_{k,c})$. The functions $\tilde{f}_{k,c}$ is the value functions that the algorithm uses to approximate the optimal value functions $v_{k,c}$. Note that in the algorithm only $\tilde{f}_{k,c}$ need to be computed at the vertices of the small simplexes and $\hat{f}_{k,c}$ is for ease of notation. Finally, let

$d_{k,c}(\vec{x})$ be an element of Σ maximizing the right-hand side of 2. It's not hard to see that only polynomially many point-value pairs (in the number of states, the horizon and the approximation precision ϵ) are to be stored in order to find near optimal actions during the course of action execution.

The following claim, shown by an induction on k and using the *Lipschitz*-1 properties of the functions $v_{k,c}$, establishes the approximation claim:

Claim 2.4 *[BRS96]* $\|v_{k,c} - \tilde{f}_{k,c}\| \leq k\delta$, *for all* $k \geq 0$ *and* $1 \leq c \leq |C|$.

To summarize, the algorithm keeps track of piecewise linear value functions on X for each color c, for each stage up to K. These functions are used to approximate optimal k step value functions, $1 \leq k \leq K$. The authors, using the *Lipschitz*-1 properties of the optimal value functions $v_{k,c}$, show that the increase in approximation error from step $k-1$ to k remain within appropriate bounds, so that the approximation result holds.

2.6 Discussion

Given the hardness of POMDPs, it is unlikely that exact methods without any assumption of extra structure in a POMDP problem will be efficient in practice. Therefore it is advisable to look for plausible structural properties and settle for approximation algorithms that take advantage of such properties and do well on problems with those properties. One structural property is the connectivity of the underlying graph representation of the system state and transitions. This property applies to MDPs as well and I will address it in the next section. We saw that introducing partial observability, uncertainty over the current system state, makes the MDP problem computationally intractable as the hardness results of 2.4 suggest. Hence imposing natural restrictions on partial observability can be promising. The work we went over is one example of such an approach wherein the uncertainty about the current state is over a bounded number of states (the *bounded-state* model).

There are other interesting restrictions on partial observability. Consider situations where each state has its own signal (color), but with some probability (the error probability) the signaler may malfunction and for instance may not emit any thing (or emit the color of another state). In a POMDP model, with criteria such as above (finite horizon, reach a target state), are there efficient approximation algorithms if we bound the error probability from above by a constant (e.g. at least half the time the signaler works)? Call such a POMDP model the *bounded-error* model. An example is given in Figure 2a: each state (circled) emits its true color half of the time, and a generic color (the question mark) the rest of the time. The bounded-error restriction can be appropriate for modeling unreliable signalers or sensors. There are other plausible models with restrictions on unobservability: we may assume for example that information on the current state could be "late," it is received after a fixed $k \geq 0$ steps of execution (a *bounded-lag* restriction).

We may also want to consider combining POMDP models with restrictions on partial observability to get models which yield more efficient algorithms. The approximation algorithm in Section 2.5 runs in exponential time as a function of color multiplicity. Is there an approximation algorithm with a lower run time complexity if in addition we also assume the problem is a *bounded-error* one? Figure 2b shows a combination of the bounded-error and bounded-state models. For POMDPs with large state spaces, it may be the case that different portions of the state space may exhibit different structural properties, each with their own efficient approximation algorithms. Can these algorithms be combined to solve large problems efficiently? For systems on which we have some control on choice of sensor capabilities, knowledge of tradeoffs in partial observability and efficient computation of optimal or near optimal strategies may help us in a better system design (see Section 4.3 for a similar situation).

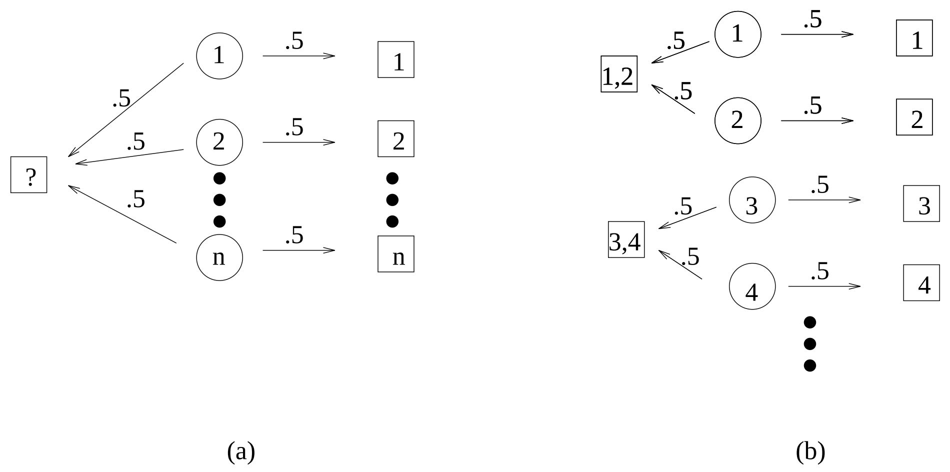

Figure 2: (a) Each state has its own color (distinguished by the numbers in the squares) but the color signaler has a fixed (independent of n) error rate of 0.5. (b) A combination of the bounded-error and the bounded-state models. At each execution step, the uncertainty is on at most two of states, and half the time there is no uncertainty.

An important question is whether real world problems with a POMDP flavor exhibit special structures like the kinds mentioned above. The work in [BRS96] was motivated by a robot navigation problem. Unfortunately, I don't have information on whether the model was useful for the problem and whether the algorithm was implemented. Works on a similar problem of robot navigation [KGS95] and medical decision making [Hau97] do not reference this work, and rely on heuristics or worst-case exponential time exact algorithms to solve their problems.

To conclude, there remains several interesting open problems with regards to finding plausible restricted POMDP problems that yield to efficient approximation. A major motivating factor for studying these problems further would be real world problems that can be satisfactorily solved utilizing such models.

3 Simple Stochastic Games

Simple stochastics games (SSGs) are a special kind of two player stochastic games, also referred to as *Markov games*, introduced by Shapley [Sha53]. Numerous variations of Shapley's model have been studied and many algorithms have been proposed (see [Con93] for references). Condon in [Con92] investigates the computational complexity of a decision problem associated with SSGs.

A significant motivation for studying the SSG model is best stated in [Con93]: "it is the simplest possible restriction of Shapley's games which retains just enough complexity so that no polynomial time algorithm for the problem is known." As we will see, the study of SSGs further motivates investigation of outstanding open problems regarding algorithms for MDPs (the one player game versions). From a computational complexity point of view, the SSG value problem, to be defined below, is complete for the class of languages accepted by log-space bounded alternating Turing machines that can also choose moves randomly (see [Con92] for details). If the value problem is in P, then randomizing does not add (significant) power to log-space bounded alternating machines: such alternating Turing machines already accept exactly the languages in P. From an application viewpoint, a variation on SSGs, called mean-payoff games in [ZP96], arise naturally in the analysis of worst case behavior of algorithms for several on-line problems, for example finite-window string matching and metrical task systems.

In the following sections, I will define the problem, explain in some detail the path to showing that the problem lies in $NP\cap$ co-NP, and conclude with a discussion of proposed algorithms and related open problems.

3.1 Problem Definition

A *simple stochastic game* (SSG) is a directed graph $G = (V, E)$ with three kinds of vertices (states), min, max, and average along with two sink vertices, the 0-sink and the 1-sink (Figure 3a). Let V_{min} and V_{max} be the set of min and max vertices, respectively. Each vertex has two outgoing edges (possibly to the same vertex), except for the sink vertices, which have no outgoing edges. One of the vertices is a start vertex s. The game is a contest between two players, player 0 and player 1. The game begins by placing a token on the start vertex s (the initial state). When the game is on a max vertex i, player 1 moves it along one of the outgoing edges of vertex i. When the game is on a min vertex j, player 0 moves it along one of the outgoing edges of j. When the token is on an average vertex, the edge along which the token is moved to is determined by the toss of a fair coin. The game ends when one of the sink vertices is reached. Player 1 wins if the token reaches the 1-sink, and player 0-wins otherwise, that is, if the token reaches the 0-sink or the game never ends.

Just as in Section 2.3, the question of what strategies are sufficient for optimality arises here too. Informally, a strategy is a rule that tells a player which edge to pick when it is the player's turn to play. The rule may be randomized and it may use any information about the state and the history of the game. However, here we restrict outselves to pure stationary and positional strategies to be defined below. Condon in [Con92] shows that for a restricted but important class of SSGs, we do not lose generality in restricting to these types of strategies.

A strategy σ of player 1 is one in which for each max vertex i the player always chooses the same edge whenever the token is placed on that vertex. A strategy τ of player 0 is defined in a similar way. Such a strategy is called pure because the player need not pick the edge randomly and it is called stationary because the choice of the edge is dependent only on the vertex and does not change with other information such as how the vertex is reached or the number of game steps played up to that point. Notice that these strategies are positional in the sense of Section 2.3 as well, where the states of the system are the vertices.

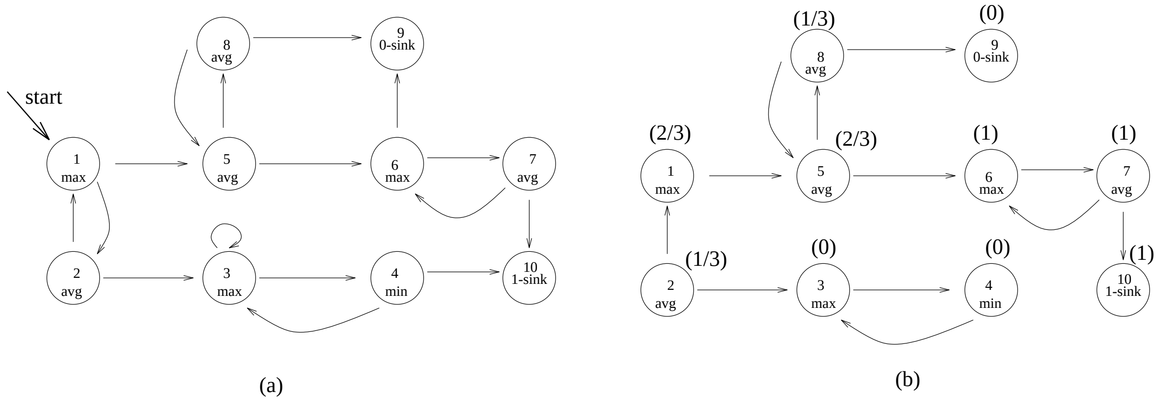

Figure 3: (a) A simple stochastic game. (b) Optimal values and an optimal pair of strategies for the game in (a).

We define the value $v_{\sigma,\tau}(i)$ of each vertex i of G with respect to strategies σ and τ to be the probability that player 1 wins the game (i.e. the token reaches the 1 sink) if the start vertex is i and the players use the strategies σ and τ. We define the value of the game G to be $\max_\sigma \min_\tau v_{\sigma,\tau}(s)$. Informally, the value of the game is the maximum probability that player 1 wins if it reveals its best strategy to player 0 at the start of the game, and player 0 plays its best strategy against the strategy chosen by player 1. Note that since the number of strategies is finite, we can use max and min instead of sup and inf.

The SSG value problem is the decision problem, given a SSG, is its value greater than 1/2? Of course, another problem is to find an optimal strategy σ^* for player 1 so that $\max_\sigma \min_\tau v_{\sigma,\tau}(s) = \min_\tau v_{\sigma^*,\tau}(s)$.

3.2 Membership in $NP \cap$ co-NP

In this section, I will describe several important and conceptually simplifying properties of SSGs proven in [Con92]. A notable property is the fact that both players possess “optimal” strategies that guarantee the best possible outcome regardless of the start vertex or the other player's choice of strategy. Condon uses these properties to show that the SSG value problem is in $NP \cap$ co-NP.

Corresponding to a strategy σ for player 1 is a graph G_σ, which is the subgraph of G obtained by removing from each max vertex the outgoing edge that is not in the strategy σ. Similarly, corresponding to a pair of strategies σ and τ (for the 1 and 0 players respectively) is a graph $G_{\sigma,\tau}$, obtained from G_σ by removing from each min vertex the outgoing edge that is not in the strategy τ. Note that $G_{\sigma,\tau}$ can be considered a Markov chain. We say that a SSG halts with probability one if for all pairs of strategies σ and τ, $G_{\sigma,\tau}$ has a path to a sink vertex. Lemmas 3.1, 3.2, and 3.3 are proved in [Con92] for SSGs that halt with probability 1 for simplicity, and I report them in their general form here. Some of the results have appeared in earlier literature on games (see for example [PV87]) and Condon includes them for completeness and, in some cases, proves stronger versions of them for SSGs.

Recall that in the SSG value problem, we are interested in the value of the start vertex of the graph. For a strategy σ of player 1, it is conceivable that any strategy of player 0 that minimizes the value of the start vertex may depend on the start vertex. The next lemma shows that this is not the case. For any strategy σ of player 1, there is some strategy of player 0 that achieves the best possible at any min vertex.

Lemma 3.1 *Let G be any SSG. Let σ be any strategy of player 1. There is some strategy τ of player 0 such that for each vertex $i \in V_{min}$ with neighbors j and k, $v_{\sigma,\tau}(i) = \min[v_{\sigma,\tau}(j), v_{\sigma,\tau}(k)]$.*

Let us denote a strategy τ with the above property by $\tau(\sigma)$, *i.e.* for each vertex $i \in V_{min}$ with neighbors j and k, $v_{\sigma,\tau(\sigma)}(i) = \min[v_{\sigma,\tau(\sigma)}(j), v_{\sigma,\tau(\sigma)}(k)]$. It is not hard to show, using some monotonicity properties, that in fact $\tau(\sigma)$ is optimal with respect to σ, *i.e.* $v_{\sigma,\tau(\sigma)}(i) = \min_\tau v_{\sigma,\tau}(i), \forall i \in V$.

The fact that both players have "optimal" strategies is a consequence of the next lemma.

Lemma 3.2 *Let G be any SSG. Then there is a strategy σ of player 1 such that, for some optimal strategy $\tau = \tau(\sigma)$ of player 0 with respect to strategy σ, for all vertices $i \in V_{max}$, with neighbors j and k, $v_{\sigma,\tau}(i) = \max[v_{\sigma,\tau}(j), v_{\sigma,\tau}(k)]$*

It is not hard to show using Lemma 3.2 that for any SSG, there is a strategy σ for player 1 which is the best that player 1 can do, in the sense that player 1 need not change its strategy assuming that player 0 plays one of its best strategies against strategy σ of player 0. In that sense σ is optimal for player 1, and Lemma 3.2 states that at least one such strategy exists for player 1 for any SSG. A similar result holds for player 0. A pair of strategies σ and τ with the above properties are called a *pair of optimal* strategies.

Finally Condon shows that the following minimax property holds for games that halt with probability 1. Similar results had been proven before for various stochastic games models. However, the restriction to pure strategies (for SSGs) is new in her paper:

Lemma 3.3 *Let G be any SSG that halts with probability 1. Then for any vertex i of G, $\max_\sigma min_\tau v_{\sigma,\tau}(i) = min_\tau \max_\sigma v_{\sigma,\tau}(i)$.*

An n vector $\vec{x}$ is called a *fixed-point* of a game SSG G if

$$\vec{x}_i = \begin{cases} \max\{\vec{x}_j, \vec{x}_k\}, & \text{if } i \text{ is a max vertex of } G \text{ with neighbors } j,\ k, \\ \min\{\vec{x}_j, \vec{x}_k\}, & \text{if } i \text{ is a min vertex of } G \text{ with neighbors } j,\ k, \\ 1/2(\vec{x}_j + \vec{x}_k), & \text{if } i \text{ is an average vertex of } G \text{ with neighbors } j,\ k, \\ 0, & \text{if } i \text{ is the 0-sink,} \\ 1, & \text{if } i \text{ is the 1-sink,} \end{cases}$$

Think of the functions $v_{\sigma,\tau}$ as n-vectors as well, so that $v_{\sigma,\tau}(i)$ is the ith coordinate of the corresponding vector. Lemma 3.2 shows that for any optimal pair of strategies σ and $\tau(\sigma)$, $v_{\sigma,\tau}$ is a fixed-point of G when G halts with probability 1.

In general, a SSG may not have a unique fixed-point. However, a *stopping* SSG does. A stopping SSG is a SSG such that with any move of either player there is some nonzero probability that the game ends in the 0-sink before it gets to the next non-average vertex. Clearly a stopping SSG halts with probability 1 (hence the above lemmas apply). Furthermore, Shapley in [Sha53] shows that stopping SSGs have unique fixed-points[3].

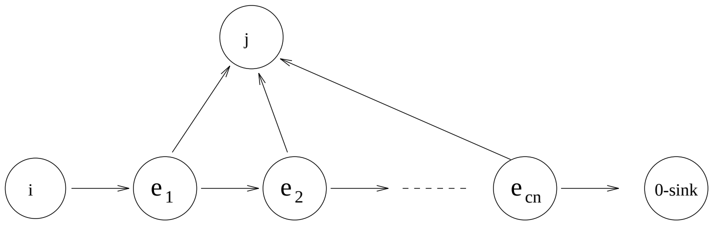

Figure 4: Adding extra average vertices to convert to a stopping game.

Condon shows that any SSG G can be converted in polynomial time to a special kind of a stopping SSG G' such that the value of G is greater than 1/2 if and only if the value of the corresponding stopping game G' is greater than 1/2. The basic conversion method is to replace each edge (i, j) of G by cn many average vertices in the way shown in Figure 4, where c is a constant (5 is provably enough) and n is the number of vertices of G.

The following property is one more important step for showing that the problem lies in both NP and co-NP.

Lemma 3.4 *[Con92] The value of a simple stochastic game with n vertices is of the form p/q, where p and q are integers, $0 \leq p, q \leq 4^{n-1}$.*

The following theorem follows easily from the above properties of stopping SSGs and the conversion procedure mentioned above:

Theorem 3.5 *The SSG value problem is in $NP\cap$ co-NP.*

Proof. Construct the $1/2^{cn}$-stopping game G' as outlined above. G' is a stopping game, so it must have a unique fixed-point, and by Lemma 3.2 the values of the vertices are the entries of

[3]This result holds for SSGs that halt with probability 1 in general.

the unique fixed-point vector for G'. Guess a $\vec{x}$ where each component of $\vec{x}$ is rational in the range given in Lemma 3.4, and verify that $\vec{x}$ is a the fixed-point of G'. If so and if the value of the start vertex, $\vec{x}(s)$, is greater 1/2 then accept, else reject. For membership in co-NP do the same except output the opposite. □

3.3 Open Algorithmic Problems

In the previous section, we went over several properties of SSGs, eventually establishing that the SSG value problem is in $NP\cap$ co-NP. Next, I will describe some of the algorithmic aspects of the problem and touch on interesting open problems. I will first place SSGs in the context of MDPs and Shapley's stochastic games.

Note that the SSG model is basically a special two player infinite-horizon MDP, where states are called vertices, and there are two actions available to each player. Condon in [Con92] shows that the problem remains in $NP \cap coNP$ with natural extensions to the model such as presence of multiple edges (actions) at each vertex, extending the transition probabilities to be rational fractions other than 1/2 or associating rewards with edges or vertices and modifying the objective criteria for each player accordingly (see also [ZP96]). Hence we note that simple stochastic games, are a generalization of fully observable MDPs where there are two decision makers instead of one. In Shapley's games model, the players simultaneously choose an action at each state of the game. SSGs are a special case because at each state at most one of the players has more than one alternative. It is not hard to see that in Shapley's general case, optimal strategies may need to be randomized (mixed), so these games are probably more complex to solve. The value of a simultaneous stochastic game need not be rational (see [PV87]), and as a consequence, the corresponding value problem is unlikely to be in $NP \cap coNP$.

SSGs have just enough complexity so that no polynomial time algorithm for the problem is known. If we restrict the states of the SSG to be only max and random or min and random, then we get a one person game or the classical infinite-horizon MDP problem solvable in polynomial time by use of linear programming. Without loss of generality, assume the SSG is a stopping one. Then the following linear program does the job: minimize $\sum_{i=1}^{n} x_i$ subject to the constraints

$$\begin{array}{ll} x_i \geq x_j & \text{if } i \text{ is a max vertex of } G \text{ with neighbor } j, \\ x_i \geq 1/2(x_j + x_k) & \text{if } i \text{ is an average vertex of } G \text{ with neighbors } j,\ k, \\ x_i = 0 & \text{if } i \text{ is the 0-sink}, \\ x_i = 1 & \text{if } i \text{ is the 1-sink}, \\ x_i \geq 0 & 1 \leq i \leq n. \end{array} \tag{3}$$

The solution to the linear program is the unique fixed-point of the game, which, by the results of the previous section, is sufficient for finding the value of the game and an optimal strategy for the player.

If we restrict the vertices to be only max and min, that is if we take the stochastic nature of the game out, then the problem is solvable in polynomial time and a simple algorithm for such appears in [Con92] (also follows from an algorithm in [ZP96]).

Next, I will describe an algorithm for SSGs, called the *Hoffman-Karp* algorithm [HK66], which is based on the strategy improvement method attributed to Howard and Bellman for infinite-horizon MDPs (see [Put94]). The algorithm is conceptually simple, and can be thought of as a local search in the space of strategies. It is a good candidate for being a polynomial time algorithm for SSGs.

We assume that the SSG given is a stopping one so that a unique and optimal fixed-point value function exists. Given a pair of strategies σ and τ, we say that a vertex i with children j and k is $v_{\sigma,\tau}$ switchable if when i is a max node $v_{\sigma,\tau}(i) < \max(v_{\sigma,\tau}(j), v_{\sigma,\tau}(k))$, and if i is a min node $v_{\sigma,\tau}(i) > \min(v_{\sigma,\tau}(j), v_{\sigma,\tau}(k))$. Switching a switchable vertex means changing the choice of its outgoing edge.

```
algorithm Hoffman-Karp
    let $\sigma$ and $\tau$ be arbitrary max and min strategies, respectively.
  repeat
      let $\sigma'$ be obtained from $\sigma$ by switching all $v_{\sigma,\tau}$ switchable vertices.
      let $\tau' \leftarrow \tau(\sigma')$.
      let $\sigma \leftarrow \sigma', \tau \leftarrow \tau'$
  until $v_{\sigma,\tau}$ is the fixed-point.
```

Note that finding $\tau(\sigma')$ is solving a one player game and can be achieved in polynomial time by a solving a linear program similar to the one in 3. The correctness of the algorithm follows from the fact that $v_{\sigma',\tau'}(i) \geq v_{\sigma,\tau}(i)$ and if i is a switchable max node, the inequality is strict. Surprisingly, slight variations in the algorithm can make the algorithm incorrect. For example either of the following two changes can cause the algorithm to go into an infinite loop on some graph instances: (1) instead of switching in the first assignment of loop, we modify the algorithm so that $\sigma' = \sigma(\tau)$, *i.e.* σ' is optimal with respect to τ (this would seem to be an improvement!), or (2) instead of finding $\tau'(\sigma')$, switch all min vertices that are switchable. Counterexamples showing that the resulting algorithms and several other plausible algorithms are incorrect can be found in [Con93].

It is not known whether the Hoffman-Karp algorithm finds optimal strategies in polynomial time or there are graphs and starting points where it can take exponentially many iterations to converge. Interestingly, a similar question can be asked of the strategy improvement algorithm of Howard for (infinite-horizon) MDPs. Assume there are only max and average vertices in the graph, and that G is stopping. The strategy improvement algorithm for the one player game is basically the above algorithm with τ or σ fixed:

```
algorithm strategy-improvement
    let $\sigma$ be an arbitrary max strategy.
  repeat
```

```
        let $\sigma'$ be obtained from $\sigma$ by switching all $v_\sigma$ switchable vertices.
        let $\sigma \leftarrow \sigma'$
    until $v_\sigma$ is the fixed-point.
```

This algorithm has excellent performance in practice. However [MC94] show that restrictions on the algorithm where only one switchable vertex is switched at each iteration[4] can go through exponentially many strategies before finding an optimal one. Does the nonrestricted ("parallel" switching) strategy improvement avoid this?

A subexponential randomized algorithm for SSGs was proposed by Ludwig in [Lud95]. Ludwig adapts methods used by Kalai [Kal92] who develops a subexponential randomized simplex method for linear programming. In Ludwig's algorithm, if G has no *max* vertices, then an optimal strategy for player 0 can be computed in polynomial time. Otherwise, a max vertex u is uniformly chosen at random, and one of the edges e from u is picked. The original graph G is reduced to $\tilde{G} = (\tilde{V}, \tilde{E})$, where $\tilde{V} = V - \{u\}$, and $\tilde{E} := E - \{(i,j) | i = u$ or $j = u\} \cup \{(i,j) | (i,u) \in E$ and $e = (u,j)\}$. The algorithm is applied recursively to $\hat{G}$ to get a pair of optimal strategies τ and σ. If τ and σ with addition of e form an optimal pair for G the algorithm stops. Otherwise the other edge from u must be optimal, so the graph is reduced by one vertex, and the algorithm repeats. The algorithm runs in $O(2^{\sqrt{n}}) \times poly(n)$ expected time. Ludwig's algorithm could probably be improved if information from one iteration could be used in later iterations.

It is perhaps the case that both the stochastic aspects of the game and the the 2-player aspects are not understood well enough to devise polynomial time algorithms for the problem. The deterministic version of SSGs (*i.e.* no average vertices) with weights, rewards or costs, associated with edges is relatively hard too. Zwick and Paterson investigate a version of deterministic 2-player games called *mean-payoff* games which arise in the study of some online problems [ZP96]. These are basically the deterministic version of SSGs with edges having payoff (rewards or costs) associated with them and one player (player 1) wants to maximize the average payoff obtained per move over the infinite horizon, while the adversary (player 0) wants to minimize it. Again, the vertex set is partitioned into two subsets of max vertices where player 1 chooses an edge, and min vertices where player 0 chooses an edge. The authors reduce their problem to SSGs and devise a pseudo-polynomial time dynamic programming-based algorithm running in $O(n^3|E||W|)$ for finding the infinite horizon values of the vertices, where $|W|$ is the maximum edge reward. The algorithm finds $v_k(i)$, by using dynamic programming k many times, each iteration taking $O(|E|)$ time. They show that $k = 4n^3|W|$ iterations is sufficient and necessary (by showing an example) to find the average costs of each vertex using their algorithm.

Condon investigates several correct algorithms for SSGs in [Con93] including one based on a quadratic program. Although any of the proposed algorithms in [Con93] and others for 2 player games may run in polynomial time, it would perhaps be more insightful to take a more direct

[4]Several vertex selection strategies are investigated.

approach to solving these games models that would utilize more of the structural properties of MDPs, specifically the graph theoretic properties. [PT87] note that the problem of deriving a "clean" polynomial time algorithm for MDPs, without using general linear programming or approximate techniques, is an important open question that has not been emphasized in the literature. Current methods do not take advantage of the MDP directed graph representation. In deterministic MDPs, utilizing graph properties (finding cycles, shortest paths, etc.) is relatively natural and polynomial time graph-theoretic algorithms are known that are a function of the number of states and actions only. Similar to the case of moving from stochastic to deterministic transitions in MDPs, the mean-payoff game restricted to one player has strongly polynomial time algorithms because the problem reduces to finding minimum or maximum mean weight cycle in a graph. We should note however that both the (stochastic) MDP problem and the two player game, when payoffs are constants (0 or 1 say), are P-complete, whereas the deterministic MDP problem and the one player mean-payoff game are in NC (see [PT87] and [Lit96]), so the deterministic versions are very likely strictly easier. We may still ask whether graph theoretic properties and methods based on them should be abandoned when we move from one player to two players in mean payoff games or from deterministic transitions to stochastic transitions in MDPs. It is the presence of cycles in SSGs that makes the analysis of algorithm for them relatively hard. SSGs with no cycles can be solved in polynomial time using dynamic programming.

We see that there are a few related open problems regarding existence of algorithms for the infinite-horizon MDP problems and their variations. For example, strategy improvement may run in polynomial time for MDPs, and in that case the insights in the analysis may also show that the Hoffman-Karp algorithm is a polynomial time algorithm for SSGs. Condon suggests that solving the following problem may be fruitful: Consider two classes of SSGs, one is the set of all SSGs where the value of start vertex is greater than 3/4 and another where the value is less than 1/4. Is there a polynomial time algorithm that distinguishes the two classes? A slightly more general requirement would be to seek a polynomial time approximation in a sense of approximation similar to subsection 2.5. Can Hoffman-Karp, with polynomially many iterations on n or the length of the input, or finite-horizon dynamic programming algorithms, with a horizon polynomial in n or the length of the input, serve as polynomial time approximation algorithms for SSGs? A similar question can be asked for approximating MDPs as well, which might be simpler to answer and can provide important insights.

4 Discrete Event Systems

We saw in the previous problems that the main source of change in the modeled systems are one or more decision makers. System or environmental influences on state changes can only be modeled through the presence of probabilistic effects in state transitions. Here we briefly study another extreme, the discrete event system (DES), where state transitions are caused by events occurring independent of the decision maker. We say that the events are system generated. The role of the decision maker in this case is mostly limited to a supervisor, influencing state transitions only through "disabling" some subsets of the events, the so-called controllable events, at various states of the system. In general, the control objective is to preserve desirable system properties and can range from preventing unwanted sequences of events in order to ensure safety (e.g. avoid dangerous states) to liveness (i.e. guarantee a minimum level of performance).

I will cover two control problems in this area. The first appears in [RW87] which introduced the formalism of DES. The second appearing in [ML97] has a stochastic flavor and is closer to MDPs. The main goal here is to understand the concepts behind the modeling, how they arise, and some of the issues involved, in the course of which differences with the above models and possible ways of combining the models may suggest themselves.

4.1 The Abstract System

In this report we assume the underlying system is an automaton, though other models such as petri nets and process algebras have also been investigated. We are given a system $G = (Q, \Sigma, d, s)$ traditionally referred to as a *plant* in control theory literature. G is an automaton: Q is a set of states, Σ is a finite set of symbols called the *event* alphabet, $d : Q \times \Sigma \rightarrow Q$ is a partial state transition function (i.e. $d(i, a)$ may not be defined), s is the initial state. We view the system G as a "generator" of sequences of events: Extend d to a function on strings in the usual way, and let $L(G) = \{w \in \Sigma^*, d \text{ defined on } w\}$ denote the set of any sequence of events that can be "generated" by G. We should view $L(G)$ as the set of "physically" possible sequences of events.

4.2 The Supervisory Synthesis Problem

The general problem is to find a supervisor (controller) that influences the behavior of the plant in such a way that it meets the objectives of the control problem. Practical control problems for which control of discrete event systems has been analyzed include database operations and communication networks.

In this section we assume that we are given the specification of a legal or permissible sequence of events in terms of the set of strings accepted by an automaton E. The language accepted by the automaton E, $L(E)$, is exactly the set of permissible sequence of events. Without loss of generality, we may assume $L(E) \subseteq L(G)$. In the paper [RW87], the sought after supervisor is also a language recognizer (a finite automaton here), a natural choice, since we want the supervisor to recognize some subset of sequences of event. The means of influence of the supervisor is blocking or permitting events. An event a is blocked in state i of the supervisor if the transition function of the supervisor is undefined for a at state i. In many situations, some events are simply uncontrollable, and if the supervisor does not want such an event to occur in a given state, then it has to make sure that the state is not reached. To model controllable and uncontrollable events, the event set Σ is partitioned into two sets of controllable, Σ_c, and uncontrollable, Σ_u, events. We assume that the supervisor cannot block an uncontrollable event.

The control problem, referred to as the supervisory control synthesis, is thus: Given the system G, and a specification E of the desired behavior, synthesize an automaton $S = (Q_S, \Sigma, d_S, s_0)$, called a supervisor, such that the composition of S with G (the product automaton) denoted by $S \| G$ here, behaves as desired, i.e. $L(S\|G) = L(E)$, where $L(S\|G)$ is the set of strings of events on which the transition function for the composite automaton $S\|G$ is defined.

The following example appearing in [RW87] should make a few of the concepts more concrete. Consider two users of a single resource, each modeled as in Figure 5a. The combined system which is the generator G to supervise is shown in Figure 5b. Here, state 0 is the start state, which corresponds to both users being their *Idle* state. The objective of supervisory control is to enable or disable the controllable events b_1 and b_2 ($\Sigma_c = \{b_1, b_2\}$) in order to satisfy the following synchronization requirement:

1. Mutual exclusion: g_1 and g_2 never simultaneously occupy their respective *Use* states.

2. Fair Usage: The *Use* states of g_1 and g_2 are occupied according to the first-come-first-served discipline.

It is not hard to verify that the desired behavior $L(E)$ is realized by the automaton in figure 5.c. Hence there is a supervisor S such that $L(S\|G) = L(E)$ and, a supervisor S for which $L(S\|G) = L(E)$ is also given by the same automaton in figure 5c (However there is in fact a supervisor with only 5 states that meets the requirements). Note how event blocking occurs in the supervisor of 5c. For example in states 2 or 3 of S event b_2 cannot occur (it is blocked). Note furthermore that such blocking of events could not be specified as a function of states in G solely, without violating the fairness specification (for example, which of b_1 or b_2 if any should be blocked at state 4 of G?). Naturally, sequences of events are significant for control in this model, unlike the case with fully observable MDP problems where control need only be state dependent. On the other hand, we note that the control is state dependent in a sense, in that the supervisor looks at its own internal state (distinct from the system state) to determine whether to block an event or not.

Given a system G and a specification E, where $L(E) \subseteq L(G)$, a supervisor does not necessarily exist. It is shown in [RW87] that a supervisor exists only if the plant cannot generate a sequence of events w that is legal, followed by a sequence u composed of uncontrollable events only, that makes the whole sequence, wu, illegal, *i.e.* $wu \notin L(E)$. To formalize the characterization we need the following two definitions:

For a language K, denote by $\bar{K}$ the prefix closure of K defined as: $\bar{K} = \{w \in \Sigma^* | \exists v \in \Sigma^*, wv \in K\}$.

Definition 1 *: Let G be a plant and Σ_u the set of uncontrollable events. The language K is said to be controllable if $\bar{K}\Sigma_u \cap L(G) \subseteq \bar{K}$, where $\bar{K}\Sigma_u = \{s\sigma \in \Sigma^* | s \in \bar{k}, \sigma \in \Sigma_u\}$.*

Theorem 4.1 *Let G be a plant and E a specification of the legal behavior, with $L(E) \subseteq L(G)$. There exists a supervisor, S, such that $L(G \| S) = L(E)$ iff the language $L(E)$ is controllable.*

If the language $L(E)$ is not controllable, then there exists no supervisor such that $S \| G$ generates exactly all legal strings. But in this case, we may be content with the relaxed control objective that the system generate no illegal strings. Then a supervisor is looked for such that $L(G \| S) \subseteq L(E)$.

Theorem 4.2 *Let G be a plant and E a specification of the legal behavior, with $L(E) \subseteq L(G)$. There exists a supervisor S, such that $L(G \| S) \subseteq L(E)$ iff there exists a prefixed-closed and controllable language contained in $L(E)$.*

In fact, in this case, we rather seek a supervisor that allows a maximal set of of legal behavior without allowing any nonlegal sequence of events to occur. We can do better: [RW87] show that the set of languages that are prefix-closed, controllable and contained in $L(E)$ is closed under arbitrary unions. This implies a *unique* supremal element in that set such that the supremal element is controllable and any other controllable language that is a subset of $L(E)$ is a subset of this language. Now, since we are assuming that both $L(G)$ and specification $L(E)$ are regular languages, then from lattice theory, a fixed point polynomial time algorithm is known that computes the supremal language. The algorithm runs in polynomial time in the the size of the state sets of the automata G and E.

The theory above is of course much more extensive than what I presented. Some extensions include requiring further desirable properties from the supervisor, finding the minimum size supervisor ("efficient" synthesis), extensions of the system model (nondeterminism, petri nets, etc.) and adding timing considerations into the model. Next, I will describe a DES model that is closer conceptually to the MDP framework.

4.3 A Discrete Event Stochastic System

Maimon et al present a version of a DES system called a *Discrete Event Stochastic System* (DESS) in [ML97] which trades off the value of the information against the cost of gathering information. Here, the similarities to Markov decision processes are apparent. As we shall see, the similarities include, stochasticity in state transitions, presence of values/costs obtained for reaching some of the states, and the fact that the supervisor takes more of an active role in controlling the evolution of the system. Applications of such models arise in the design of automatic supervisors in manufacturing, safety, and maintenance problems. See below for an example application.

A DESS is modeled as a 6-tuple $G = (Q, \Sigma, p, s, Q_m, v)$, where Σ is the finite event set, Q is the finite state set. As in the previous section the event alphabet Σ is partitioned into two sets of controlled events Σ_c and uncontrollable events Σ_u, where $\Sigma_c \cap \Sigma_u = \phi$. The difference from the above is that the state set is also partitioned into Q_c, states having controlled events defined only (hence controllable), and Q_u, states having uncontrolled events defined only. p

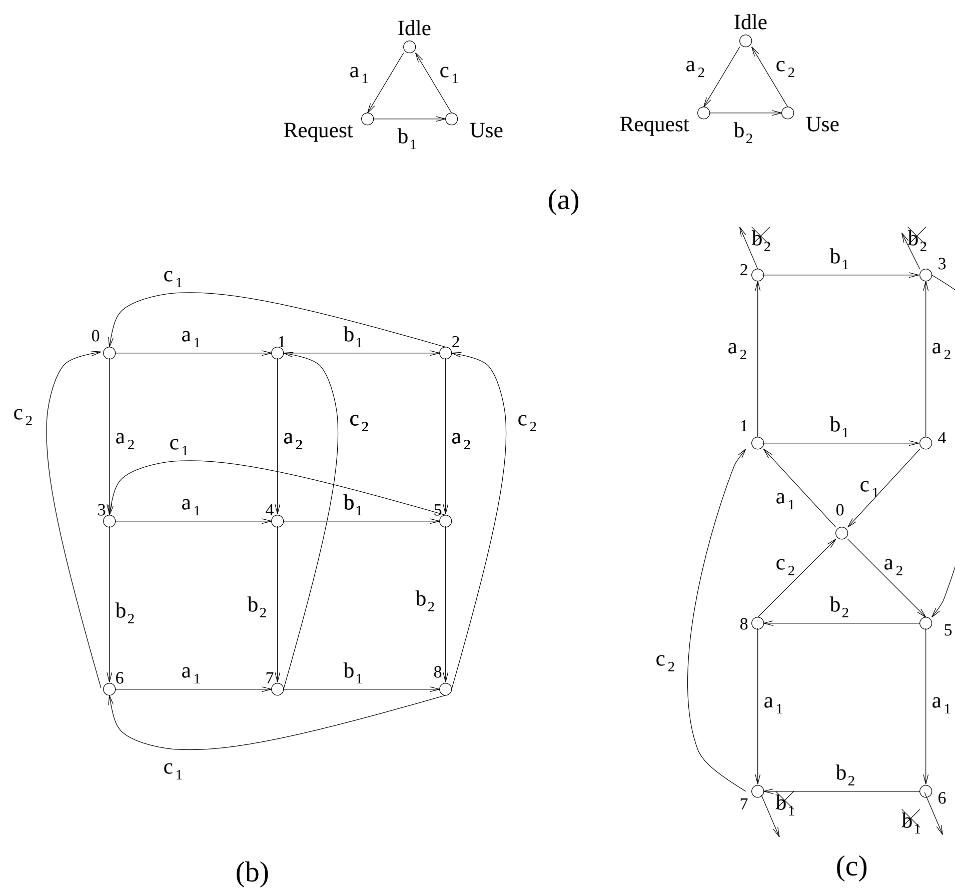

Figure 5: (a)Users of a single resource. (b) The combination of the 2 systems. (c) The specification automaton and the supervisor.

characterizes both the probability of uncontrolled event occurrences and state transitions: $p(a|q)$ is the probability of an uncontrolled event a occurring in state $q \in Q_u$, and $p(j|i,a)$ denotes the probability of a transition from state i to state j given event a, controllable or uncontrollable, occurring in state i. s is the initial state of the system. $Q_m \subset Q$ is a set of "marked" states. A sequence of activities from the start state to a marked state represents a finished cycle of activity. For $q \in Q_m$, we regard $v(q)$ as the cost charged if q is reached. The goal is to minimize expected costs subject to constraints described below.

The key concept here is the notion of the *information set.* An information set w is a subset of Q_c with the following properties:

1. w has more than one state in it.

2. All states in w have the same set of controlled events.

3. All states in w have a common uncontrollable origin state $o(w)$, such that the probability of a transition to any state of w from $o(w)$ is greater than zero, while no state out of w

can be reached directly from $o(w)$. No other state in Q has a transition to any state in w.

From the above properties, it follows that information sets cannot overlap. Therefore the number of information sets is always less then the number of states in the DESS (in fact, at most $|Q_c|/2$). An example of an information set is given in Figure 6, where there are two possible states in the information set, each having the same controllable event set $\{a, b\}$ defined on them.

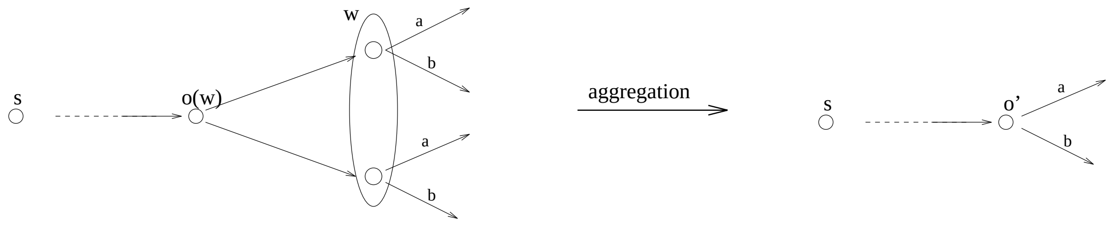

Figure 6: An information set with two states in it.

Now, the problem here is whether to treat an information set w as an *aggregate* state, or detect each state in w individually. Let $\psi(w) = 0$ when we choose to aggregate states, and $\psi(w) = 1$, otherwise, *i.e.* when the supervisor chooses to detect the states in the aggregate set individually (the case of perfect information).

In the case of aggregation on w, we can imagine replacing states in w together with $o(w)$ with one controllable state o' with the same set of controllable events defined for states of w. Assume event a is defined on $q_i \in w$, and let $p(o(w), q_i)$ denote the probability of a transition from $o(w)$ to q_i. Then $p(j|o', a) = \sum_{q_i \in w} p(o(w), q_i)p(j|q_i, a)$.

An information vector $\vec{F}$ denotes the vector of choices $\psi(w_i)$ on the information states w_i: $\forall i : F_i = \psi(w_i)$. Note that when we fix an information vector $\vec{F}$, the problem of finding the best controllable events (actions) becomes an MDP problem, and an optimal strategy σ^* can be computed in polynomial time. For a state $q \in Q_m$, denote by $p_{\sigma^*}(q|\vec{F})$ the probability that state q is reached from the initial state q_0, if the information vector $\vec{F}$ is used, and an optimal control σ^* is applied, *i.e.* optimal events are chosen on any controllable state that is not aggregated. The expected cost of the system denoted by $EC(\vec{F})$ is defined to be $EC(\vec{F}) = \Sigma_{q \in Q_m} v(q).p_{\sigma^*}(q|\vec{F})$.

The problem of designing an information efficient supervisor can then be stated as follows:

$$\min EC(\vec{F}), \tag{4}$$

subject to the constraint that $\Sigma_{i=1}^{m} y_i F_i \leq K$, where m is the number of information sets, y_i is a weight factor which is a cost measure for obtaining perfect information on information set w_i, and K is an upper-bound on the total cost of sets observed prefectly in $\vec{F}$.

I will briefly go over one example application appearing in [ML97] to further illustrate the concepts. In [ML94] an aircraft maintenance application is presented.

Imagine a robot that is in charge of testing several barrels of chemicals containing dangerous material. The danger is that uncontrolled sequence of chemical events can evolve suddenly and rapidly that may lead to explosion and damage. There are costs associated with leaving a barrel in dangerous state and also removing a barrel. The robot has at its disposal two main types of measurement: one being a crude but fast and cheap measurement action and another more elaborate but more costly measurement. There are constraints on the frequency of performing the second measurements. The problem is after measurement 1 (whose outcome is an uncontrolled event) whether to take measurement 2 (the perfect information case), and then choose the best control action (leave or take out a barrel) based on the result, or choose the best control action without performing measurement 2 (aggregation). Given the event probabilities and costs associated with the different outcomes, the problem is when to take measurement 2 as a function of the outcome of measurement 1.

Problem 4 is clearly in NP, and the authors claim that the problem is NP-complete in [ML94] (no proof is provided) and present a heuristic algorithm to solve the problem. The question of existence of appropriate approximation algorithms is open.

5 Discussion

Figure 7 gives an overview of the MDP related family of problems that appeared in this report. As expected, the problems get harder computationally with generalizations: The deterministic MDP problems can be solved using graph based algorithms such as shortest path or minimum weight cycles depending on the objective criteria, while, on another extreme, infinite horizon POMDPs are undecidable in general. We discussed the relations between the game models and MDPs in section 3. Research on alternative algorithms for deterministic 2 player games (mean payoff) and for (stochastic) MDPs, might prove useful not only for SSGs, but also for POMDPs.

In many cases, problems may best be approached by using a combination of models. The SSG model can be thought of as a combination of the stochastic MDP and the deterministic 2 player games. We covered several possible restrictions to partial observability in POMDPs and conjectured that presence of multiple types of restrictions might lead to more efficient algorithms. The original discrete event systems of [RW87] seemed to be an entirely different model of control due to its assumptions and objective criteria. However, with the introduction of stochastic transition and a more active control, the DESS model obtained was basically a variation on the MDP problem: The problem to solve, due to constraints on monitoring a system, was which MDP system to choose from a set of feasible MDPs implicit in the problem statement, so that expected costs are minimized. Other hybrid models of discrete event systems and MDPs are conceivable and it is plausible that in many applications modeling both external events and active control may be more appropriate than considering either model in isolation. Real applications are needed to justify the study of potential hybrid systems in the abstract.

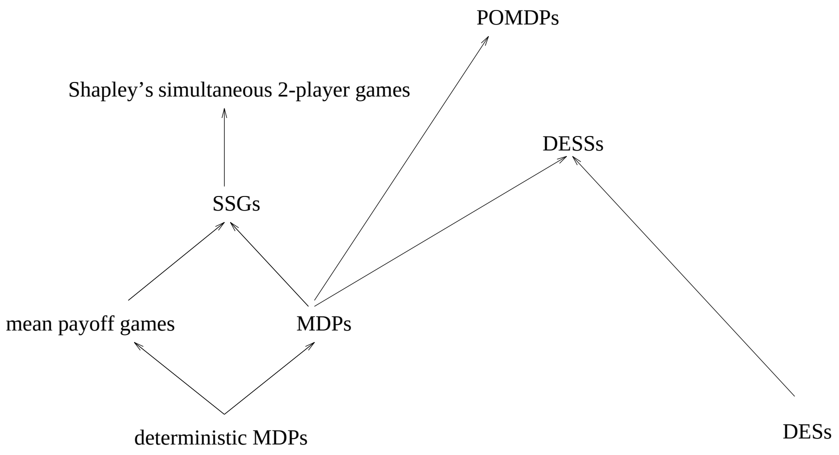

Figure 7: A family of MDP related problems.

Likewise, models of 2 player or multiplayer stochastic games where players have imperfect information on the state of the game (*e.g.* consider Shapley's simultaneous stochastic games with partial observability) are conceivable, but the existence of real applications of such models is questionable. We should note however that the investigation of the combined models may lead to a better understanding of interactions of the multi-player aspects of the games and the partial observability aspects, and in addition superior algorithm may be discovered.

6 Conclusions

There remain many interesting open problems regarding problems of control that have an MDP flavor. Generalizations, such as 2-player games, motivate the study of simpler one player MDPs more closely. This could lead to new algorithmic or analytic techniques. POMDPs have been shown to be hard to solve in general. Can we restrict the partial observability without losing applicability and at the same time obtain efficient algorithms? We saw that this is a possiblility and described one approach in Section 2. Finally, there are models of discrete control that emphasize the occurrence of state changes which are not controllable. Possibilities exist for hybrid models as described in subsection 4.3 in case of a DESS system.

While many of the remaining open problems are theoretically well-motivated, and answers to some may spawn new applications of their own, effort should also be spent on looking at existing potential application areas for a source of new challenges and to further justify the study of model variations, especially in the case of restricted POMDPs.

7 Acknowledgments

I am grateful to the many proof readers. Of course, the responsibility for any of the shortcomings in the paper rests entirely on me.

References

[BDH95] Craig Boutilier, Thomas Dean, and Steve Hanks. Planning under uncertainty: structural assumptions and computational leverage. In *New Directions in AI Planning*, pages 157–171, 1995.

[BRS96] Dima Burago, Michel De Rougemont, and Anatol Slissenko. On the complexity of partially observed Markov decision processes. *Theoretical Computer Science*, pages 161–183, 1996.

[Con92] Anne Condon. The complexity of simple stochastic games. *Information and Computation*, 96(2):203–224, February 1992.

[Con93] Anne Condon. On algorithms for simple stochastic games. In *Advances in computational complexity theory*, volume 13 of *DIMACS series in discrete mathematics and theoretical computer science*. 1993.

[Dan91] George B. Dantzig. Linear programming. In J.K. Lenstra, A.H.G. Rinnooy Kan, and A. Schrijver, editors, *History of Mathematical Programming, A Collection of Personal Reminiscences*, page 30. 1991.

[DW91] Thomas L. Dean and Michael Wellman. *Planning and Control*. Morgan Kaufmann, 1991.

[Hau97] Milos Hauskrecht. Dynamic decision making in a stochastic partially observable medical domain: Ischemic heart disease example. In *Artificial Intelligence in Medicine*, Lecture Notes in Artificial Intelligence, pages 296–299. 1997.

[HK66] A. Hoffman and R. Karp. On nonterminating stochatic games. *Management Science*, 12(5), 1966.

[Kal92] Gil Kalai. A subexponential randomized simplex algorithm. In *24th annual ACM STOC*, 1992.

[KGS95] Sven Koenig, Richard Goodwin, and Reid Simmons. Robot navigation with markov models: A framework for path planning and learning with limited computational resources. In *Reasoning with Uncertainty in Robotics*, Lecture Notes in Artificial Intelligence, pages 322–337. 1995.

[Lit96] Michael Littman. *Markov Decision Processes and Reinforcement learning.* PhD thesis, Brown, 1996.

[Lov91] W. Lovejoy. A survey of algorithmic methods for partially observable markov decision processes. *Annals of Operations Research*, pages 47–66, 1991.

[Lud95] W. Ludwig. A subexponential randomized algorithm for the simple stochastic game problem. *Information and computation*, 117:151–155, 1995.

[MC94] Mary Melekopoglou and Anne Condon. On the complexity of the policy improvement algorithm for markov decision processes. *ORSA Journal on Computing*, 6(2), 1994.

[ML94] Oded Maimon and Mark Last. Information efficient robotic control. *Robotica*, 12:157–163, 1994.

[ML97] Oded Maimon and Mark Last. Information-efficient control of discrete event stochastic systems. *IEEE transactions on Systems, Man, and Cybernetics*, 27(1):23–32, January 1997.

[Mon82] George Monahan. A survey of partially observable markov decision processes: Theory, models, and algorithms. *Management Sceience*, 28(1):1–15, 1982.

[NM80] John Von Neumann and Oskar Morgenstern. *Theory of games and economic behavior.* Princeton University Press, 3rd edition, 1980.

[PT87] Christos H. Papadimitriou and John N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of operations research*, 12(3):441–450, August 1987.

[Put94] Martin L. Puterman. *Markov Decision Processes.* Wiley Inter-science, 1994.

[PV87] H Peters and O. Vrieze. Surveys in game theory and related topics. In *CWI Tract 39.* Amsterdam, 1987.

[RW87] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal on Control and Optimization*, 25(1):206–230, 1987.

[Sha53] L.S. Shapley. Stochastic games. *Proceedings of the National Academy of sceinces USA*, 39:1095–1100, 1953.

[SS73] R. Smallwood and E. Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations Research*, 21:1071–1088, 1973.

[Whi88] Douglas J. White. Further real applications of Markov decision processes. *Interfaces*, 18:55–61, September 1988.

[ZP96] Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *Theorteical Computer Science*, 158(1-2):343–359, May 1996.