A Hybrid Genetic/Optimization Algorithm for Finite Horizon Partially Observed Markov Decision Processes

Alex Z.-Z. Lin, James C. Bean, Chelsea C. White, III

February 26, 1999

Abstract

The partially observed Markov decision process (**POMDP**) is a generalization of a Markov decision process that allows for noise corrupted and costly observations of the underlying system state. The value function of the finite horizon **POMDP** is known to be piecewise affine and convex in the probability mass vector over the state space. Such a function can be represented by a finite set of affine functions.

In this paper, we develop and evaluate an exact algorithm, **GAMIP**, which integrates a genetic algorithm and a mixed integer program to construct the minimal set of affine functions that describes the value function. Numerical results indicate that **GAMIP** takes up to 60% less time than the most efficient linear programming-based exact solution method in the literature to construct the minimal set.

1 Introduction

This paper presents a new procedure, **GAMIP**, for constructing an exact solution to the finite horizon partially observed Markov decision process (**POMDP**) with finite state, action, and observation sets. **GAMIP** makes use of a genetic algorithm (**GA**) and a mixed integer program (**MIP**) and is based on the fact that the solution of the optimality equation for the **POMDP** is piecewise affine and convex (**PAC**) in the probability mass vector (**pmv**) over the state space.

The **POMDP** is an extension of the Markov decision process (**MDP**) [27, 28, 31] which considers situations such as: state observations are costly; sensors that measure the current state value are noise-corrupted; at least part of the current state value is inaccessible. The **POMDP** requires information regarding how observations are related to state and action values and hence is more data intensive than the **MDP**. Current exact numerical solution procedures for the **POMDP** are presented in [2, 3, 4, 5, 15, 18, 19, 21, 25, 26, 30].

This paper is organized as follows. In Section 2, the **POMDP** is defined and several fundamental results are presented on which our optimal solution approach is based. A discussion of how we integrate **GA** and **MIP** is presented in Section 3. Section 4 briefly reviews the random keys **GA**, dc-niche [17], for the **GA**-step. We also discuss a new fitness evaluation function that differs from the function defined in [17]. Section 5 presents the generic **MIP** model and its enhancement. The numerical evaluation in Section 6 compares **GAMIP** to a linear programming based algorithm [4]. The results indicate that **GAMIP** consistently performs better than the **LP** method with respect to the total solution time. Section 7 summarizes research results and discusses future research directions.

2 Partially Observed Markov Decision Processes

2.1 Definition

Let **S**, **A**, and **Z** be finite sets representing the *state*, *action*, and *observation* spaces, respectively. Let $\{0, 1, \ldots, T-1\}$ be the set of *stages* (or *epochs*), where the *problem horizon* T is finite ($T < \infty$). Assume the *state* process $\{s(t),\ t = 0, 1, \ldots, T\}$, *action* process $\{a(t),\ t = 0, 1, \ldots, T-1\}$, and *observation* process $\{z(t),\ t = 1, 2, \ldots, T-1\}$ are linked by the probabilities $\mathbf{P}(z, a) = \{p_{ij}(z, a)\}$, where

$$p_{ij}(z, a) = \mathrm{P}[s(t+1) = j, z(t+1) = z \mid s(t) = i,\ a(t) = a].$$

Thus, if the underlying state of the process is $s(t)$ and the action selected is $a(t)$ at stage t, then the underlying state makes transition to state $s(t+1)$ and $z(t+1)$ is observed at stage $t+1$, according to the probability kernel $\mathbf{P}(z,a)$.

Let $r(i,a)$ be the reward accrued at stage t, given $s(t)=i$ and $a(t)=a$, and let $\bar{r}(i)$ be the reward received at terminal stage T, given $s(T)=i$. We assume the action at stage t is selected on the basis of the *history* at stage t, which is comprised of an a priori **pmv** on $\mathbf{S}$, all past and present observations, and all past actions selected. A function mapping the set of histories at stage t into the action set is a *policy* at stage t; a sequence of policies, one for each stage $t=0,1,\ldots,T-1$, is a *strategy*. The problem *objective* is to determine a strategy that maximizes the total expected discounted reward to be accrued over the problem horizon, relative to the set of all strategies, conditioned on the a priori **pmv**.

2.2 Preliminary Results

Let $\mathbf{X}=\{\mathbf{x}:\ x_i\geq 0,\ i\in\mathbf{S},\ \sum_{i\in\mathbf{S}}x_i=1\}$. Define $x(t)=\{x_i(t),\ i\in\mathbf{S}\}$, where $x_i(t)$ is the probability that $s(t)=i$, given the history at stage t. It is shown in [25, 30] that the resulting process, the *information* process $\{\mathbf{x}(t),\ t=0,1,\ldots,T\}$, is a *sufficient statistic* for the **POMDP**. Hence, it is sufficient to base action selection at stage t on $\mathbf{x}(t)$. Let $v_{t,T}(\mathbf{x})$ be the maximum expected total discounted reward to be accrued from stage t through the terminal stage, given $\mathbf{x}(t)=\mathbf{x}$. It has been shown in [30] and elsewhere that the following optimality equation and boundary condition hold:

$$\begin{aligned} v_{t,T}(\mathbf{x}) &= \max_{a\in\mathbf{A}}\left\{\mathbf{x}\mathbf{r}(a)+\beta\textstyle\sum_{z\in\mathbf{Z}}\sigma(z,\mathbf{x},a)v_{t+1,T}[\lambda(z,\mathbf{x},a)]\right\}, \\ v_{T,T}(\mathbf{x}) &= \mathbf{x}\bar{\mathbf{r}}, \end{aligned} \tag{1}$$

where $\sigma(z,\mathbf{x},a)=\mathbf{x}\mathbf{P}(z,a)\underline{\mathbf{1}}$, $\lambda(z,\mathbf{x},a)=\mathbf{x}\mathbf{P}(z,a)/\sigma(z,\mathbf{x},a)$ $(\sigma(z,\mathbf{x},a)\neq 0)$, and $\beta\geq 0$ is the discount factor. We remark that $\lambda(z,\mathbf{x},a)$ is the **pmv** at stage $t+1$ when $z(t+1)=z$, $\mathbf{x}(t)=\mathbf{x}$, and $a(t)=a$, and $\sigma(z,\mathbf{x},a)$ is the probability of observing z at stage $t+1$, given that $\mathbf{x}(t)=\mathbf{x}$ and $a(t)=a$. An optimal strategy is composed of policies which for stage t selects an action that causes the maximum to be attained in (1) when $\mathbf{x}(t)=\mathbf{x}$ [30]. Hence, solving

the optimality equation for $t = 0, 1, \ldots, T$ is key to determining an optimal strategy. We now examine structural results that lead to the solution of the optimality equation.

2.3 Structural Results

If $v_{t+1,T}$ is **PAC**, then $v_{t,T}$ is **PAC**, and since $v_{T,T}$ is **PAC**, $v_{t,T}$ is **PAC** for all t [30]. We remark that if a real-valued function f on $\mathbf{X}$ is **PAC**, then there exists a finite set Γ such that $f(\mathbf{x}) = \max\{\mathbf{x}\gamma : \gamma \in \Gamma\}$. We wish to determine $v_{t,T}$ from $v_{t+1,T}$ by determining $\Gamma_{t,T}$, given $\Gamma_{t+1,T}$, where $v_{t,T}(\mathbf{x}) = \max\{\mathbf{x}\gamma : \gamma \in \Gamma_{t,T}\}$ and $v_{t+1,T}(\mathbf{x}) = \max\{\mathbf{x}\gamma : \gamma \in \Gamma_{t+1,T}\}$. The set $\Gamma_{t,T}$ can be constructed from $\Gamma_{t+1,T}$ as follows. Define $\mathbf{G}(\Gamma) = \bigcup_{a\in\mathbf{A}} \left\{\mathbf{r}(a) + \beta \sum_{z\in\mathbf{Z}} \mathbf{P}(z,a)\gamma^z : \ \gamma^z \in \Gamma\right\}$, and note that

$$\begin{aligned} v_{t,T}(\mathbf{x}) &= \max_{a\in\mathbf{A}} \left\{\mathbf{x}\mathbf{r}(a) + \beta \sum_{z\in\mathbf{Z}} \sigma(z,\mathbf{x},a) \max\left[\lambda(z,\mathbf{x},a)\gamma : \ \gamma \in \Gamma_{t+1,T}\right]\right\} \\ &= \max_{a\in\mathbf{A}} \left\{\mathbf{x}\mathbf{r}(a) + \beta \sum_{z\in\mathbf{Z}} \max\left[\mathbf{x}\mathbf{P}(z,a)\gamma : \ \gamma \in \Gamma_{t+1,T}\right]\right\} \\ &= \max\left\{\mathbf{x}\gamma : \ \gamma \in \mathbf{G}(\Gamma_{t+1,T})\right\}. \end{aligned}$$

2.4 PURGE Operator

Thus, we could set $\Gamma_{t,T} = \mathbf{G}(\Gamma_{t+1,T})$, or more generally, we could set $\Gamma_{t,T}$ to be any subset of $\mathbf{G}(\Gamma_{t+1,T})$ satisfying the condition $\max\{\mathbf{x}\gamma : \gamma \in \Gamma_{t,T}\} = \max\{\mathbf{x}\gamma : \gamma \in \mathbf{G}(\Gamma_{t+1,T})\}\ \forall\ \mathbf{x} \in \mathbf{X}$. However, from both storage and computational perspectives, there is value in keeping the cardinality of $\Gamma_{t,T}$ as small as possible. Let γ be called *redundant* (in Γ on $\mathbf{X}$) if and only if for all $\mathbf{x}' \in \mathbf{X}$, there is a $\gamma' \in \Gamma$ such that $\gamma' \neq \gamma$ and $\mathbf{x}'\gamma' \geq \mathbf{x}'\gamma$. Thus, we wish to remove as many redundant vectors from $\mathbf{G}(\Gamma_{t+1,T})$ as possible in order to define $\Gamma_{t,T}$.

Let the operator **PURGE** be such that $\Gamma' = \mathbf{PURGE}(\Gamma)$ is the subset of Γ having the smallest cardinality that satisfies $\max\{\mathbf{x}\gamma : \ \gamma \in \Gamma'\} = \max\{\mathbf{x}\gamma : \gamma \in \Gamma\}$ for all $\mathbf{x} \in \mathbf{X}$. Each element in set Γ' is referred to as a *defining* vector. Existence of such a subset is assured by results in [18]. Then ideally we would want to select $\Gamma_{t,T} = \mathbf{PURGE}(\mathbf{G}(\Gamma_{t+1,T}))$. Define

1. For two sets Γ^1 and Γ^2, let $\Gamma^1 \oplus \Gamma^2 = \{\gamma^1 + \gamma^2 : \gamma^1 \in \Gamma^1,\ \gamma^2 \in \Gamma^2\}$.

2. For scalar β and set Γ, let $\beta\Gamma = \{\beta\gamma :\ \gamma \in \Gamma\}$.

3. For vector γ' and set Γ, let $\gamma' + \Gamma = \{\gamma' + \gamma :\ \gamma \in \Gamma\}$.

Then, $\Gamma_{t,T} = \mathbf{PURGE}(\mathbf{G}(\Gamma_{t+1,T}))$ can be determined as follows [29, 32]:

$$\Gamma^{z,a}_{t+1,T} = \mathbf{PURGE}\left(\{\mathbf{P}(z,a)\gamma :\ \gamma \in \Gamma_{t+1,T}\}\right), \tag{2}$$

$$\Gamma^{a}_{t,T} = \left\{\mathbf{r}(a) + \beta\mathbf{PURGE}\left(\bigoplus_{z\in\mathbf{Z}} \gamma^z :\ \gamma^z \in \Gamma^{z,a}_{t+1,T}\right)\right\}, \tag{3}$$

$$\Gamma_{t,T} = \mathbf{PURGE}\left(\bigcup_{a\in\mathbf{A}} \Gamma^{a}_{t,T}\right). \tag{4}$$

Let $''\circ''$ be either the set addition operator defined above or the union operator; i.e., $\circ \in \{\oplus, \cup\}$. It is straightforward to show that for $\circ \in \{\oplus, \cup\}$ and for $K \geq 2$,

$$\mathbf{PURGE}(\Gamma^1 \circ \cdots \circ \Gamma^K) = \mathbf{PURGE}(\mathbf{PURGE}(\Gamma^1 \circ \cdots \circ \Gamma^{K-1}) \circ \Gamma^K),$$

where the sets $\Gamma^k, k = 1, \ldots, K$, are assumed given. These properties suggest an exact algorithm to construct $\Gamma_{t,T}$, known as *incremental pruning* [32]. The pseudocode is summarized in Figure 1. We remark that the most efficient algorithms reported [4, 32] are based on this approach.

```
                  Γ_{t,T} ← ∅.
                  For a ∈ A {
                      Γ^a_{t,T} ← ∅.
                      For z ∈ Z {
Form 1                    Γ^{z,a}_{t+1,T} ← PURGE({P(z,a)γ : γ ∈ Γ_{t+1,T}}).
Form 2                    Γ^a_{t,T} ← PURGE(Γ^a_{t,T} ⊕ Γ^{z,a}_{t+1,T}).
                      }
Form 3                Γ_{t,T} ← PURGE(Γ_{t,T} ∪ Γ^a_{t,T}).
                  }
```

Figure 1: Pseudocode for incremental pruning

We remark that the cardinality of $\Gamma^{z,a}_{t+1,T}$ is usually much smaller than the cardinality of $\Gamma_{t+1,T}$, and hence the **PURGE** step described in (2) is especially useful in removing redundant

vectors. As noted in [8], a procedure that has proven effective in removing redundant vectors in $\{\mathbf{P}(z,a)\gamma : \gamma \in \Gamma\}$ is: remove γ'' from Γ if there is a $\gamma' \in \Gamma$ such that $\gamma'_i \geq \gamma''_i$ for all $i \in \mathbf{S}$.

2.5 The Operator $\mathcal{A}$

We now present and analyze an operator closely related to **PURGE**. For any $\mathbf{X}' \subseteq \mathbf{X}$, define $\mathcal{A}(\Gamma, \mathbf{X}') = \{\mathcal{A}(\Gamma, \mathbf{x}') : \mathbf{x}' \in \mathbf{X}'\}$, where $\mathcal{A}(\Gamma, \mathbf{x}') = \gamma'$ if and only if (i) $\gamma' \in \Gamma$ and (ii) $\mathbf{x}'\gamma' \geq \mathbf{x}'\gamma \; \forall \; \gamma \in \Gamma$ where ties are broken lexicographically [18]. Note that $\mathcal{A}(\Gamma, \mathbf{X}) = \mathbf{PURGE}(\Gamma)$ and $\mathcal{A}(\Gamma, \mathbf{X}') \subseteq \mathcal{A}(\Gamma, \mathbf{X})$.

2.6 Determination of $\mathcal{A}(\Gamma^1 \circ \Gamma^2, \mathbf{x}')$

We observe that the three forms of the **PURGE** operator given in Figure 1 are related to the determination of $\mathcal{A}(\Gamma^1 \circ \Gamma^2, \mathbf{x}')$, given Γ^1, Γ^2, and $\mathbf{x}'$. The following lemma presents two useful results which are extended from [18].

Lemma 1 *Let $\gamma^i = \mathcal{A}(\Gamma^i, \mathbf{x}')$, $i \in \{1, 2\}$. Then*

1. $\gamma^1 + \gamma^2 = \mathcal{A}(\Gamma^1 \oplus \Gamma^2, \mathbf{x}')$.

2. $\gamma^i = \mathcal{A}(\Gamma^1 \cup \Gamma^2, \mathbf{x}')$ *if and only if* $\mathbf{x}'\gamma^i \geq \mathbf{x}'\gamma^j$, $j \in \{1, 2\}$, $j \neq i$.

We remark that determination of $\mathcal{A}(\Gamma, \mathbf{x}')$, given Γ and $\mathbf{x}'$, is straightforward. In general, we seek a finite subset $\mathbf{X}' \subseteq \mathbf{X}$ such that $\mathcal{A}(\Gamma, \mathbf{X}') = \mathcal{A}(\Gamma, \mathbf{X})$. In this paper, we will use a **GA**, the first step of **GAMIP**, to determine $\mathbf{X}''$, the set of so-called *witness points*, so that $\mathcal{A}(\Gamma, \mathbf{X}'')$ is a good approximation of $\mathcal{A}(\Gamma, \mathbf{X})$. We then use a **MIP** to discover additional witness points to add to $\mathbf{X}''$ in order to produce $\mathbf{X}'$ so that $\mathcal{A}(\Gamma, \mathbf{X}') = \mathcal{A}(\Gamma, \mathbf{X})$.

2.7 Determination of $\mathcal{A}(\Gamma, \mathbf{X})$

Assume $\mathcal{A}(\Gamma, \mathbf{X}'')$ is given for finite set of witness points $\mathbf{X}'' \subseteq \mathbf{X}$. The algorithm in Figure 2 determines $\mathcal{A}(\Gamma, \mathbf{X}') = \mathcal{A}(\Gamma, \mathbf{X})$. Note that since $\Gamma' \subseteq \Gamma$, $v(\mathbf{x}) \geq z(\mathbf{x})$ and $u^* \geq 0$. See

[2, 3, 4, 5, 15, 18, 19, 21, 25, 26, 30] for more details of **LP**-based algorithms for determining $\mathcal{A}(\Gamma, \mathbf{X})$ from Γ and $\mathcal{A}(\Gamma, \mathbf{X}')$.

Step 0. Set $\Gamma' = \mathcal{A}(\Gamma, \mathbf{X}'')$ and $\mathbf{X}' = \mathbf{X}''$.

Step 1. Determine the $\mathbf{x}' \in \mathbf{X}$ that causes the maximization in the following problem to be attained:

$$\begin{array}{rcll} u^* & \equiv & \text{MAX} & \{v(\mathbf{x}) - z(\mathbf{x})\} \\ & & \text{S.T.} & v(\mathbf{x}) = \max\{\mathbf{x}\gamma : \gamma \in \Gamma\} \\ & & & z(\mathbf{x}) = \max\{\mathbf{x}\gamma : \gamma \in \Gamma'\} \\ & & & \mathbf{x} \in \mathbf{X}. \end{array} \tag{5}$$

Step 2a. If $u^* > 0$, then determine $\gamma' \in \Gamma$ such that $\mathbf{x}'\gamma' \geq \mathbf{x}'\gamma \ \forall \gamma \in \Gamma$. Add γ' to Γ', $\mathbf{x}'$ to $\mathbf{X}'$, and go to **Step 1**.

Step 2b. If $u^* = 0$, then set $\mathcal{A}(\Gamma, \mathbf{X}) = \Gamma'$ and stop.

Figure 2: Pseudocode for the determination of $\mathcal{A}(\Gamma, \mathbf{X})$ from $\mathcal{A}(\Gamma, \mathbf{X}')$

3 Solution Approach: GAMIP

Recall that we are interested in developing an exact algorithm to determine $\Gamma_{t,T}$ for the **POMDP**, given $\Gamma_{t+1,T}$. Adopting the incremental pruning method depicted in Figure 1 and the properties discussed in Section 2.4, the core issue of an efficient optimal algorithm is to construct the set Γ' from the set Γ efficiently such that $\Gamma' = \mathbf{PURGE}(\Gamma)$. We use a three-step backward recursive procedure in Figure 3, **GAMIP**, in constructing the desired set Γ'.

GA-step. Find a subset of the desired Γ-set (denoted as $\Gamma^{\mathbf{GA}}$) which is as large as possible. We have developed a **GA**-based technique that accomplishes this step quickly.

DOM-step. Perform the componentwise domination procedure [8] on each element in Γ against $\Gamma^{\mathbf{GA}}$ to quickly remove redundant vectors in Γ.

MIP-step. Find all remaining γ-vectors in the desired Γ-set that were not found in the **GA**-step. This step is often considerably more computationally intensive than the **GA**-step.

Figure 3: The three steps for the **GAMIP** approach

The more γ-vectors are found in the **GA**-step and more redundant γ-vectors are removed in the **DOM**-step, the less CPU time is required to perform the **MIP**-step.

The use of a **GA** is motivated by two observations:

1. It is easy to find a defining γ-vector in Γ', given Γ and an $\mathbf{x}' \in \mathbf{X}$.

2. GA can be quite effective in rapidly finding good suboptimal solutions [1, 11, 12, 22, 23, 24].

An approximation of the minimal Γ-set, $\Gamma^{\mathbf{GA}}$, is constructed by finding the *defining* γ-vectors associated with those witnesses generated by the **GA**. The purpose of the **DOM**-step is to reduce the cardinality of the set Γ under consideration and hence the size of the problems solved in the **MIP**-step. We use a **MIP** in the **MIP**-step to solve (5) in order to either prove that $\Gamma^{\mathbf{GA}}$ is exactly Γ', or find the defining γ-vector(s) missed by the **GA** and thus to determine the optimal solution.

To enhance the performance of **GAMIP**, we can either improve the effectiveness of the **GA**-step, which will result in the decrease of the number of **MIP**s to be solved, or reduce the size of **MIP**s to be solved in the **MIP**-step. We remark that **Form 2 PURGE** (see Figure 1) is the most computationally intensive step in the incremental pruning approach. The next two sections instantiate **GAMIP**, detailing a specialized **GA**-search, dc-niche, and a **MIP**, to

exactly determine $\Gamma' = \mathbf{PURGE}(\Gamma^1 \oplus \Gamma^2)$. See [16] for instantiations of **GAMIP** on the other two **PURGE** operations.

4 Genetic Algorithm

Genetic algorithms, introduced in the mid-60's [13], solve complex problems by a paradigm from biological evolution. A population of solutions is constructed. A next generation is derived from the old population with operations that promote *survival-of-the-fittest*. Over many generations the solutions in the population improve until the best of the population is (hopefully) near optimal. In this paper and [17], we are interested in using a **GA** to determine a set of witness points which can be used to construct a good approximation of the value function for the finite horizon **POMDP**. In [17] we detail **GA**s used as a heuristic approach for the **POMDP**. Here, we refine one of those **GA**s and extend it to an optimal algorithm with the **MIP**. Two random keys **GA**s, dc-niche and spatial [17], are developed and have been shown to construct good value function approximations. We adopt the random keys **GA** dc-niche as a module for our exact algorithm in this paper.

In general, there are several key components to a **GA** approach: encoding a solution, fitness evaluation function, reproduction processes, and stopping criteria. Except for the definition of fitness evaluation function, all instantiations of these fundamental aspects of dc-niche can be found in [17]. We now briefly review the basic construction of dc-niche.

4.1 Basics of Random Keys GA: dc-niche

A *chromosome* is an encoding of a solution and is a vector of variables in $\mathcal{R}^n$. A *gene* is an element of that chromosome vector; an *allele* is a numerical value taken by a gene. In this paper and [17], the following encoding scheme is taken. Let the i^{th} gene of the chromosome denote the probability that the system is in state i. Since each chromosome is a distribution, a feasible assignment of alleles must sum to one. This requirement creates issues with offspring feasibility

that are resolved with the use of random keys [1].

The random keys **GA** essentially works with two spaces: search space and solution space. All the **GA** operators act upon the search space. Each point in search space is transformed into a feasible solution and evaluated in the solution space. In our context, we use the random keys **GA** to search a majority of a positive unit hypercube. Each point in this hypercube is transformed into a feasible point in the information state space $\mathbf{X}$ by normalization. These normalized points in $\mathbf{X}$ are then used to determine the approximate Γ-set, $\Gamma^{\mathbf{GA}}$, for describing the value function of finite horizon **POMDP**.

A **GA** generates new points in the reproduction process in order to yield near optimal solutions. As indicated in [17], since the function to optimize in dc-niche is multi-modal, niching methods [6, 7, 9, 10, 13, 14, 20], in particular the crowding scheme [7, 20], help this process in generating as many new witness points as possible. For more details see [17].

4.2 Fitness Evaluation Function

We now determine the real-valued fitness evaluation function (**FEF**) that will be used to measure the fitness of a chromosome. We will develop a new **FEF** for the optimal algorithm relative to the heuristic in [17]. As noted in [17], an **FEF** for the **POMDP** should measure the difference between a value function and its approximation with the intent of improving the quality of the approximation. We restrict our attention to **PAC** value functions and approximations, and hence restrict our attention to sets containing a finite number of γ-vectors. The function $\mathbf{F}$ defined in [17] is:

$$\mathbf{F}(\mathbf{x}, \Gamma', \Gamma^{\mathbf{GA}}) \quad = \quad \left\{ \begin{array}{ll} f(\mathbf{x}, \Gamma', \Gamma^{\mathbf{GA}}) & \text{if } f(\mathbf{x}, \Gamma', \Gamma^{\mathbf{GA}}) > 0 \\ \Delta(\mathbf{x}, \Gamma^{\mathbf{GA}}) & \text{if } f(\mathbf{x}, \Gamma', \Gamma^{\mathbf{GA}}) = 0 \end{array} \right\} \tag{6}$$

where

1. Γ' is the γ-vector set sought.

2. $f(\mathbf{x}, \Gamma', \Gamma^{\mathbf{GA}}) = \max_{\gamma \in \Gamma'} [\mathbf{x}\gamma] - \max_{\gamma \in \Gamma^{\mathbf{GA}}} [\mathbf{x}\gamma]$.

3. $\Delta(\mathbf{x}, \Gamma^{\mathbf{GA}}) = \max^2_{\gamma \in \Gamma^{\mathbf{GA}}}[\mathbf{x}\gamma] - \max_{\gamma \in \Gamma^{\mathbf{GA}}}[\mathbf{x}\gamma]$, and

4. $\max^2_{\gamma \in \Gamma^{\mathbf{GA}}}[\mathbf{x}\gamma]$ is the second-best value attaining by $\Gamma^{\mathbf{GA}}$ at $\mathbf{x}$.

As noted in [17], use of $\mathbf{F}$ as the fitness evaluation function generates two types of local maxima: those with fitness value less than or equal to 0 and those with fitness value strictly greater than 0. The first type will not give new defining vectors and are referred to as *fruitless* attractors. The second type gives new defining vector(s) and are referred to as *fruitful* attractors.

Figure 4 illustrates this function $\mathbf{F}$ for a 2-state problem. Assume $\Gamma^{\mathbf{GA}} = \{\gamma^1, \cdots, \gamma^5\}$, the thick solid lines, and $\Gamma' = \{\gamma^1, \cdots, \gamma^9\}$ is the minimal set of affine vectors to define the value function considered. We remark that there is no need to construct Γ' explicitly in order to compute $\mathbf{F}(\mathbf{x}, \Gamma', \Gamma^{\mathbf{GA}})$. Then the dotted lines together with the four fruitful attractors depict the function $\mathbf{F}$ defined in (6).

The potential difficulty with the function $\mathbf{F}$, as indicated in [17], is that the number of fruitless attractors increases as the **GA** explores the simplex $\mathbf{X}$ and identifies new defining vectors. Let $\mathbf{A}$ and $\mathbf{B}$ be two sets of γ-vectors and $\mathbf{A} \backslash \mathbf{B} \equiv \{\gamma \in \mathbf{A} : \ \gamma \notin \mathbf{B}\}$. The example depicted in Figure 4 suggests that the new **FEF** defined below is able to reduce this potential difficulty:

$$\mathcal{F}(\mathbf{x}, \Gamma \backslash \Gamma^{\mathbf{GA}}, \Gamma^{\mathbf{GA}}) = \max_{\gamma \in \Gamma \backslash \Gamma^{\mathbf{GA}}}[\mathbf{x}\gamma] - \max_{\gamma \in \Gamma^{\mathbf{GA}}}[\mathbf{x}\gamma],$$

where $\Gamma = \Gamma^1 \oplus \Gamma^2$.

Figure 4 also illustrates this new function $\mathcal{F}$ for a 2-state problem. For simplicity, assume $\Gamma = \{\gamma^1, \cdots, \gamma^{10}\}$. Thus, $\Gamma \backslash \Gamma^{\mathbf{GA}} = \{\gamma^6, \cdots, \gamma^{10}\}$, the four dashed lines and one solid line. When γ^8 is identified by an information state $\mathbf{x}$, instead of having two more fruitless attractors as it is in $\mathbf{F}$, the function $\mathcal{F}$ still has no fruitless attractor. Moreover, the *bases* of certain fruitful attractors in $\mathcal{F}$ grow wider. A base of an attractor is the subset of $\mathbf{X}$ of starting points that would lead to the attractor in a hill-climbing algorithm. An example of a base of a fruitful attractor is given in Figure 4. These observations are given in the circle in Figure 4, in which

the thick dotted lines are the landscape of $\mathbf{F}$ and the thick solid lines are the landscape of $\mathcal{F}$. We remark that γ^{10}, which is not a defining vector in either Γ' or Γ before γ^8 is identified, is now a defining vector for the value function defined by $\Gamma \backslash \Gamma^{\mathbf{GA}} = \{\gamma^6, \gamma^7, \gamma^9, \gamma^{10}\}$.

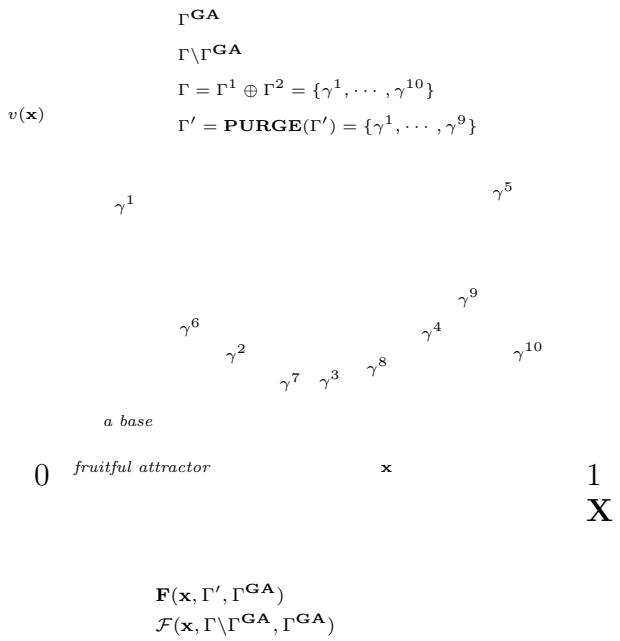

Figure 4: Illustration of two evaluation functions: $\mathbf{F}$ and $\mathcal{F}$

For implementation, the following two concerns are addressed:

1. When an undiscovered defining vector γ is identified by a witness $\mathbf{x}$, γ is removed from the set $\Gamma \backslash \Gamma^{\mathbf{GA}}$ and is added into the set $\Gamma^{\mathbf{GA}}$. This results in a change in landscape of $\mathcal{F}$ local to $\mathbf{x}$. In order to properly reflect this change in $\mathcal{F}$, the fitness of $\mathbf{x}$ needs to be recomputed with these newly updated Γ-sets.

2. For the fitness of an information state $\mathbf{x}$, the exact value of the first term in $\mathcal{F}$, $\max_{\gamma \in \Gamma \backslash \Gamma^{\mathbf{GA}}}[\mathbf{x}\gamma]$, requires the enumeration of the set $\Gamma = \Gamma^1 \oplus \Gamma^2$. For computational speed, we estimate it as follows. Let $\Gamma_2^1(\mathbf{x})$ denote all $\gamma \in \Gamma^1$ such that γ has not yielded any defining vector when paired with $\gamma^2 = \mathcal{A}(\Gamma^2, \mathbf{x})$; i.e. $\Gamma_2^1(\mathbf{x}) = \{\gamma \in \Gamma^1 : \gamma + \gamma^2 \notin \Gamma^{\mathbf{GA}}\}$. The set $\Gamma_1^2(\mathbf{x})$ and

the vector γ^1 are similarly defined. Note that both $\Gamma_2^1(\mathbf{x})$ and $\Gamma_1^2(\mathbf{x})$ are sets associated with some given information state $\mathbf{x}$. If $\gamma^1 + \gamma^2 \in \Gamma^{\mathbf{GA}}$, then it can be shown that

$$\max_{\gamma \in \Gamma \setminus \Gamma^{\mathbf{GA}}} [\mathbf{x}\gamma] \geq \max \left\{ \mathbf{x}\gamma^1 + \max_{\gamma \in \Gamma_1^2(\mathbf{x})} [\mathbf{x}\gamma],\ \mathbf{x}\gamma^2 + \max_{\gamma \in \Gamma_2^1(\mathbf{x})} [\mathbf{x}\gamma] \right\}.$$

We use the right hand side expression as an estimate of the left hand side expression. This approximation slightly lower the fitness of those information states not yielding new defining vectors but is without loss of optimality.

5 Mixed Integer Program

Recall that the problem we want to solve is formulated in (5), where $\Gamma = \Gamma^1 \oplus \Gamma^2$, $\mathcal{A}(\Gamma, \mathbf{X}') = \Gamma^{\mathbf{GA}}$, and $\mathcal{A}(\Gamma, \mathbf{X})$ is the set we seek to construct.

5.1 Basic MIP Model

Let $v^1(\mathbf{x})$, $v^2(\mathbf{x})$, and $z(\mathbf{x})$ be the maximum function values attained by the set Γ^1, Γ^2, and $\Gamma^{\mathbf{GA}}$ at point $\mathbf{x}$, respectively. It is known that all Γ^1, Γ^2 and $\Gamma^{\mathbf{GA}}$ are **PAC** representations. With the introduction of $0-1$ variables, y, (5) can be reformulated as the mixed integer program:

$$\begin{array}{rcll} u^* & \equiv & \text{MAX} & [v^1(\mathbf{x}) + v^2(\mathbf{x})] - z(\mathbf{x}) \\ & & \text{S.T.} & v^i(\mathbf{x}) \leq \mathbf{x}\gamma + (1 - y_\gamma^i)\mathbf{H},\ \forall\ \gamma \in \Gamma^i,\ i \in \{1,2\}; \\ & & & \sum_{\gamma \in \Gamma^i} y_\gamma^i = 1,\ i \in \{1,2\}; \\ & & & y_\gamma^i \in \{0,1\},\ \gamma \in \Gamma^i,\ i \in \{1,2\}; \\ & & & z(\mathbf{x}) \geq \mathbf{x}\gamma,\ \forall\ \gamma \in \Gamma^{\mathbf{GA}}; \\ & & & \mathbf{x} \in \mathbf{X}, \end{array} \tag{7}$$

where $\mathbf{H}$ is a large positive number.

The objective function is to maximize the gap between two **PAC** functions: one for the optimum, $[v^1(\mathbf{x}) + v^2(\mathbf{x})]$, and the other for the approximation, $z(\mathbf{x})$. There are three groups of constraints in (7). The first group, consisting of the first three sets of constraints, computes

$v^i(\mathbf{x})$, the maximum value attained by the set Γ^i at point $\mathbf{x}$, where $i \in \{1, 2\}$. Each binary variable y^i_γ is associated with a vector γ in Γ^i. The role of these variables and the multiple choice constraint is to ensure that there is exactly one γ-vector in Γ^i selected to define $v^i(\mathbf{x})$. The near or exact optimum attained at $\mathbf{x}$ by $\Gamma^{\mathbf{GA}}$, $z(\mathbf{x})$, is defined in the second group of constraints. The last equation in (7) ensures that the information states are in a nonnegative simplex.

If $u^* > 0$ and the information state that causes the maximum to be attained is $\mathbf{x}^* \in \mathbf{X}$, a new defining γ-vector can be obtained easily with $\mathbf{x}^*$. We remark that the **MIP** optimization processes can be stopped once a strictly positive gap (i.e., $u^* > 0$) is detected.

We may need to solve many **MIP**s in order to finalize the approximation constructed by the **GA**. Two potential difficulties can be encountered. The first difficulty is that the size of these **MIP**s can be too large to solve in a reasonable period of time. The other difficulty is that we solve a new **MIP** from scratch in each run. Next, we discuss several ideas for enhancing the performance of this **MIP** step.

5.2 Enhancement of MIP Step

To enhance the performance of the **MIP** step, we reduce the size of **MIP**s by exploiting information discovered by the **GA**s and **MIP**s, and partitioning larger **MIP**s into smaller **MIP**s.

From the **Form 2 PURGE** in Section 2.4, the optimal $\Gamma_{t,T}$ can be constructed by considering all $\gamma = \gamma^1 + \gamma^2$ for all $\gamma^1 \in \Gamma^1$ and $\gamma^2 \in \Gamma^2$ and all appropriate Γ^1, Γ^2 from $\Gamma_{t+1,T}$. For some $\gamma^1 \in \Gamma^1$ and some $\gamma^2 \in \Gamma^2$, if $(\gamma^1 + \gamma^2)$ has been shown to give either a defining vector or a redundant vector in the **GA**s, dominance tests or **MIP**s, we refer to γ^2 as a *checked candidate* with γ^1. Otherwise, we refer to γ^2 as an *unchecked candidate* with γ^1.

The full **MIP** to find missing defining vectors involves checking all γ^1, γ^2 pairs. This search can be decomposed by searching subsets of γ from $\Gamma_{t+1,T}$. So long as all combinations are eventually checked the decomposition is without loss of optimality. Due to the exponential computation of **MIP**s, it is efficient to consider a series of smaller problems rather than a single

large **MIP**. This is doubly true since information discovered in early subproblems can reduce future work.

Let $\Gamma^2_{\gamma^1} \subseteq \Gamma^2$ denote the set of unchecked candidates with $\gamma^1 \in \Gamma^1$. Then, the sub-**MIP** associated with that vector γ^1 is defined in (8). Note that the potential advantage of the **MIP** (8) over the **MIP** (7) is $|\Gamma^2_{\gamma^1}| \leq |\Gamma^2|$ for all $\gamma^1 \in \Gamma^1$.

$$
\begin{array}{rcll}
u^*_{\gamma^1} & \equiv & \text{MAX} & [\mathbf{x}\gamma^1 + v^2(\mathbf{x})] - z(\mathbf{x}) \\
& & \text{S.T.} & v^2(\mathbf{x}) \leq \mathbf{x}\gamma + (1 - y^2_\gamma)\mathbf{H},\ \forall\ \gamma \in \Gamma^2_{\gamma^1}; \\
& & & \sum_{\gamma \in \Gamma^2_{\gamma^1}} y^2_\gamma = 1; \\
& & & y^2_\gamma \in \{0,1\},\ \gamma \in \Gamma^2_{\gamma^1}; \\
& & & z(\mathbf{x}) \geq \mathbf{x}\gamma,\ \forall\ \gamma \in \Gamma^{\mathbf{GA}}; \\
& & & \mathbf{x} \in \mathbf{X}.
\end{array}
\tag{8}
$$

We can further decompose the **MIP** (8) into several smaller **MIP**s such that the maximal number of binary variables in these smaller **MIP**s is less than some prespecified number, denoted as $\mathcal{M}$.

Adopting the two steps above results in a procedure for solving the **MIP** (7) by simply partitioning the **MIP** (7) into many sub-**MIP**s as summarized in Figure 5.

$\mathcal{M} \leftarrow 15$.
while($\Gamma^1 \neq \emptyset$) {
    Select an element γ^1 from Γ^1.
    Construct $\Gamma^2_{\gamma^1} \subseteq \Gamma^2$, the set of unchecked candidates of γ^1.
    while($\Gamma^2_{\gamma^1} \neq \emptyset$) {
        Construct $\Gamma^2_{\gamma^1}(\mathcal{M})$ from $\Gamma^2_{\gamma^1}$ such that $|\Gamma^2_{\gamma^1}(\mathcal{M})| \leq \mathcal{M}$.
        Solve **MIP** (8) with $\Gamma^2_{\gamma^1}$ substituted by $\Gamma^2_{\gamma^1}(\mathcal{M})$.
        $\Gamma^2_{\gamma^1} \leftarrow \Gamma^2_{\gamma^1} \backslash \Gamma^2_{\gamma^1}(\mathcal{M})$.
    }
    $\Gamma^1 \leftarrow \Gamma^1 \backslash \{\gamma^1\}$.
}

Figure 5: The pseudocode for the decomposed **MIP**-step of **GAMIP**

6 Numerical Results

This section reports numerical evaluation results regarding the performance of two exact solutions procedures: **GAMIP** and the restricted-region approach (referred to as *lp-best*) presented in [18]. See [4, 18, 32] for the theory of lp-best.

Experiments were run on 9 examples from [5] and 5 examples from [2, 3, 17]. These 14 examples are briefly described in Table 1. Detailed data of these examples are available upon request. We assume a horizon of $T = 20$. Parameter values for the **GA**-step are in the appendix of [17]. We set $\mathcal{M} = 15$, the maximal number of $0-1$ variables defined in the **MIP**s solved in the **MIP**-step. The solution obtained by the lp-best method is the reference point of comparison.

Table 2, 3, and 4 summarize the numerical results. Table 2 summarizes the comparison of total run time in second ($\mathcal{T}$) up to horizon 20 between **GAMIP** and lp-best where δ is the ratio of computation times for **GAMIP** vs. lp-best. As indicated in the δ column of Table 2, **GAMIP** is consistently better and commonly takes half the time that lp-best requires to construct the defining Γ-set for the value function considered. The differences in the cardinality among some examples possibly result from numerical precision and the way we use **MIP** solutions to deter-

mine a defining vector. Instead of finding the defining vector attaining the maximum at the $\mathbf{x}^*$ returned by the **MIP**, we include the vector selected by $y^i_\gamma = 1$. Theoretically, this procedure would produce a slightly larger defining Γ-set than is minimal.

Table 3 details the total run times used for each of the three major steps in **GAMIP** up to horizon 20: **GA**-step, **DOM**-step, and **MIP**-step. As the results suggest, the **GA**-step uses less than 10% of the total time and the time needed to compute the **DOM**-step is negligible. Thus, the **MIP**-step consumes most computational resources.

Table 4 details the performance of **GA** in constructing an initial Γ-set on which the **MIP**-step will improve. The hit-rate of **GA** is defined as an average performance index:

$$\mathcal{H} = \frac{\sum_{i=1}^{N} \frac{|\Gamma_i^{\mathbf{GA}}|}{|\Gamma_i|}}{N} \times 100\%,$$

where at i^{th} run of **PURGE**, $\Gamma_i^{\mathbf{GA}}$ is the Γ-set constructed by the **GA** and Γ^i is the defining Γ-set, and N is the total number of **PURGE** operations that **GAMIP** needs to execute. In Table 4, four hit-rate indices are presented: $\mathcal{H}_{za}$, the hit-rate of **GA** for **Form 1 PURGE**, $\mathcal{H}_{cross}$, the hit-rate of **GA** for **Form 2 PURGE**, $\mathcal{H}_{union}$, the hit-rate of **GA** for **Form 3 PURGE**, and $\mathcal{H}_{\mathbf{GA}}$, the overall hit-rate of **GA**-step. These indices in the class of test problems are consistently high and most of them reach up to 90% or higher.

Table 1: Data setting for 14 Test Examples

Example	$\|\mathbf{S}\|$	$\|\mathbf{A}\|$	$\|\mathbf{Z}\|$	β
chengD31 [5]	3	3	3	1.00
chengD32 [5]	3	3	3	1.00
chengD33 [5]	3	3	3	1.00
chengD41 [5]	4	4	4	1.00
chengD42 [5]	4	4	4	1.00
chengD43 [5]	4	4	4	1.00
chengD44 [5]	4	4	4	1.00
chengD45 [5]	4	4	4	1.00
chengD51 [5]	5	3	3	1.00
hanks [2]	4	4	2	0.95
marking [2]	9	4	3	0.87
4x4 [3]	16	4	2	0.95
p2044 [16]	20	4	4	0.87
p3044 [16]	30	4	4	0.90

Table 2: Comparisons between **GAMIP** and lp-best algorithms

strategy	**GAMIP**		$lp-best$		δ
examples	$\|\Gamma\|$	$\mathcal{T}$	$\|\Gamma\|$	$\mathcal{T}$	$\frac{\mathcal{T}_{\mathbf{GAMIP}}}{\mathcal{T}_{lp-best}}$
chengD31	17	**5.73**	14	11.43	.50
chengD32	8	**2.23**	8	5.09	.44
chengD33	15	**12.98**	15	28.36	.44
chengD41	46	**47.40**	45	119.84	.40
chengD42	50	**42.27**	45	82.19	.51
chengD43	46	**36.52**	45	91.75	.40
chengD44	114	**398.87**	96	550.00	.72
chengD45	135	**280.79**	119	409.79	.68
chengD51	80	**65.96**	80	153.78	.43
hanks	96	**110.36**	97	211.89	.52
marking	193	**45.08**	204	93.27	.48
4x4	20	**.39**	20	11.06	.04
p2044	14	**13.50**	14	21.92	.62
p3044	124	**122.81**	124	267.14	.44
total		1184.83		2057.51	.58

Table 3: Statistics of total run-time of **GAMIP**

examples	$\mathcal{T}_{\mathbf{GA}}$	$\mathcal{T}_{\mathbf{DOM}}$	$\mathcal{T}_{\mathbf{MIP}}$	$\mathcal{T}_{\mathbf{GAMIP}}$
chengD31	1.12	.05	4.56	5.73
chengD32	.62	.01	1.60	2.23
chengD33	1.87	.00	11.11	12.98
chengD41	5.74	.22	41.44	47.40
chengD42	5.62	.18	36.47	42.27
chengD43	4.73	.10	31.69	36.52
chengD44	19.43	.83	378.55	398.81
chengD45	20.14	.85	259.80	280.79
chengD51	5.90	.25	59.81	65.96
hanks	4.46	.27	105.63	110.36
marking	4.24	.33	40.51	45.08
4x4	.33	.03	.03	.39
p2044	5.51	.14	7.85	13.50
p3044	23.15	1.03	98.63	122.81
total	102.86	4.29	1077.68	1184.83
%	8.7	0.4	90.9	100.00

Table 4: Statistics of **GAMIP**: hit-rate of **GA**-step

examples	$\mathcal{H}_{za}$	$\mathcal{H}_{cross}$	$\mathcal{H}_{union}$	$\mathcal{H}_{\mathbf{GA}}$
chengD31	.99	.94	.94	.97
chengD32	1.00	1.00	1.00	1.00
chengD33	.98	.94	.94	.96
chengD41	.98	.93	.91	.95
chengD42	.97	.92	.86	.95
chengD43	.99	.96	.91	.97
chengD44	.94	.86	.81	.90
chengD45	.95	.89	.83	.93
chengD51	.97	.91	.90	.95
hanks	.82	.83	.83	.83
marking	.97	.92	.86	.95
4x4	1.00	1.00	1.00	1.00
p2044	1.00	.97	.94	.99
p3044	.99	.97	.94	.98

7 Conclusions and Extensions

In this paper, we have developed a numerical procedure, **GAMIP**, for determining the value function and an optimal policy for a finite horizon **POMDP**. **GAMIP** is based on the integration of a **GA** and a **MIP** into an optimal procedure. Numerical results show that **GAMIP** requires less computational time than other numerical procedures. The enhanced performance of **GAMIP** is due to the high performance of the **GA**-step and the control of the size of **MIP**s solved.

The **MIP**s solved form a family of ordered **MIP**s such that the two consecutive **MIP**s differ by only one constraint in the second group of constraints relating to $z(\mathbf{x})$ in **MIP** (7) or (8). In future research, we will reformulate the **MIP** to find multiple γ-vectors in a single application of the **MIP**.

Acknowledgment: This work was supported by the National Science Foundation under Grant DMI-9634712.

References

[1] J. C. Bean. Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal of Computing*, 6 (2):154–160, Spring 1994.

[2] A. R. Cassandra. *Exact and Approximate Algorithms for Partially Observable Markov Processes*. PhD thesis, Brown University, Province, RI, 1998.

[3] A. R. Cassandra, L. P. Kaelbling, and M. L. Littman. Acting optimally in partially observable stochastic domains. Technical Report CS-94-20, Dept. of Computer Science, Brown University, April 1994.

[4] A. R. Cassandra, M. L. Littman, and N. L. Zhang. Incremental pruning: a simple, fast, exact algorithm for partially observable Markov decision processes. Technical report, Dept. of Computer Science, Brown University, Feb 1997.

[5] H.-T. Cheng. *Algorithms for partially observable Markov decision processes*. PhD thesis, University of British Columbia, British Columbia, Canada, 1988.

[6] Y. Davidor. A naturally occurring niche and species phenomenon: the model and first results. Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann, 1991.

[7] K. A. De Jong. *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan, 1975.

[8] J. N. Eagle. The optimal search for a moving target when the search path is constrained. *Operations Research*, 32 (5):1107–1115, Sep–Oct 1984.

[9] D. E. Goldberg and J. J. Richardson. Genetic algorithms with sharing for multimodal function optimization. Proc. 2nd International Conference on Genetic Algorithms, Lawrence Erlbaum Publishers, 1987.

[10] M. Gorges-Schleuter. Explicit parallelism of genetic algorithms through population structure. Parallel Problem Solving from Nature, Springer-Verlag, 1990.

[11] A. B. Hadj-Alouane and J. C. Bean. A genetic algorithm for the multiple-choice integer program. *Operations Research*, 45 (1):92–101, Jan–Feb 1997.

[12] A. B. Hadj-Alouane, J. C. Bean, and K. G. Murty. A hybrid genetic/optimization algorithm for a task allocation problem. Technical Report 93-30, Dept. of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, MI 48109-2117, November 1998, Revised.

[13] J. H. Holland. *Adaptation in Natural and Artificial Systems.* The Universiy of Michigan, 1975.

[14] J. H. Holland. *Adaptation in Natural and Artificial Systems.* Cambridge, MA: MIT press, 1992.

[15] J. W. Lark. *A heuristic search approach for solving finite horizon, completely unobserved Markov decision processes.* PhD thesis, University of Virginia, 1989.

[16] A. Z.-Z. Lin. *A Hybrid Genetic/Optimization Algorithm for a Class of Sequential Decision Models.* PhD thesis, University of Michigan, Michigan, U.S.A., 1998. in preparation.

[17] A. Z.-Z. Lin, J. C. Bean, and C. C. White, III. Genetic algorithm heuristics for finite horizon partially observed Markov decision processes. Technical Report 98-24, Dept. of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, MI 48109-2117, 1998.

[18] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling. Efficient dynamic programming updates in partially observable Markov decision processes. Technical Report CS-95-19, Dept. of Computer Science, Brown University, December 1995.

[19] W. S. Lovejoy. A survey of algorithmic methods for the partially observable Markov decision processes. *Annals of Operations Research*, 28:47–66, 1991.

[20] S. W. Mahfoud. *Niching method for genetic algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, May 1995.

[21] G. E. Monahan. A survey of partially observable Markov decision processes: theory, models, and algorithms. *Management Science*, 28:1–16, Jan. 1982.

[22] B. A. Norman and J. C. Bean. A genetic algorithm methodology for complex scheduling problems. Technical Report 94-5, Dept. of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, MI 48109-2117, July 1996. To appear in *Naval Research Logistics*.

[23] B. A. Norman and J. C. Bean. A random keys genetic algorithm for job shop scheduling. *Engineering Design and Automation*, 3:145–156, 1997.

[24] B. A. Norman and J. C. Bean. Operation scheduling for parallel machine tools. Technical Report 95-29, Dept. of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, MI 48109-2117, April 1998, Revised. To appear in *IIE Transaction*.

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

[26] E. J. Sondik. *The optimal control of partially observable Markov processes*. PhD thesis, Standford University, California, USA, 1971.

[27] D. J. White. Real applications of Markov decision processes. *Interfaces*, 15:7–83, 1985.

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

[29] D. J. White. Piecewise linear approximation for partially observable Markov decision processes with finite horizons. *Journal of Information and Optimization Sciences*, 13 (2):311–324, 1992.

[30] C. C. White, III. A survey of solution techniques for the partially observed Markov decision processes. *Annals of Operations Research*, 32:215–230, 1991.

[31] C. C. White, III and D. J. White. Markov decision processes. *European Journal of Operations Research*, 39:1–16, 1989.

[32] N. L. Zhang and W. Liu. Planning in stochastic domains: problem characteristics and approximation. Technical Report HKUST-CS96-31, Department of Computer Science, Hong Kong University of Science and Technology, 1996.