Proceedings of the
47th IEEE Conference on Decision and Control
Cancun, Mexico, Dec. 9-11, 2008

Polynomial Classification Algorithms for Markov Decision Processes

Eugene A. Feinberg* and Fenghsu Yang†

Abstract

The unichain classification problem detects whether an MDP with finite states and actions is unichain or not under all deterministic policies. This problem has been proven to be NP-hard. This paper provides polynomial algorithms for this problem while there exists a state in an MDP, which is either recurrent under all deterministic policies or absorbing under some action.

1. INTRODUCTION

We consider discrete-time Markov Decision Processes (MDPs) with finite state and action sets in this paper. The probability structure of an MDP is defined by a state space $S = \{1,\dots,N\}$, finite sets of actions $A(i)$ for all $i \in S$, and transition probabilities $p(j|i,a)$, where $i, j \in S$ and $a \in A(i)$. A deterministic policy φ is defined as a function from S to $\bigcup_{i\in S} A(i)$ which assigns an action $\varphi(i) \in A(i)$ to each state $i \in S$. Each deterministic policy defines a stochastic matrix $P(\phi) = (p(j|i,\varphi(i))_{i,j=1,\dots,N}$. The stochastic matrix defined by a deterministic policy is also known as a transition matrix of a homogeneous Markov chain. A transition matrix determines which states of the Markov chain are recurrent, transient, or equivalent.

A state $i \in S$ is called *transient* (*recurrent*) if it is transient (recurrent) under all deterministic policies. An MDP is called *multichain*, if the transition matrix corresponding to at least one deterministic policy φ contains two or more recurrent classes. Otherwise, an MDP is called *unichain*. Thus, under any deterministic policy, the state space of a unichain MDP consists of a single recurrent class plus a possible empty set of transient states.

The property of unichain is important for MDPs with average reward criterion because stronger results on the existence of optimal policies hold and better algorithms are available for unichain MDPs than for general MDPs; see [8] for detail. Unichain MDPs have been treated separately from general MDPs since Howard [6] introduced the policy iteration algorithms for MDPs; see e.g. [4, 5, 8, 10]. In 2002, Kallenberg [7] studied irreducibility, communicating, weakly communicating, and unichain classification problems for MDPs. For the first three problems, Kallenberg [7] constructed polynomial algorithms. However, for the unichain classification problem, Kallenberg [7], [8, p. 41] posted a question whether a polynomial algorithm exists. In 2007, Tsitsiklis [11] answered this question by showing that the unichain classification problem is *NP*-hard.

Even though the unichain classification problem is *NP*-hard, many applications are modelled as unichain MDPs. Moreover, many applications of MDPs contain the states which are recurrent under all stationary policies. For instance, for a queueing or inventory control problem, a recurrent state is typically either the state when the buffer is empty or the state when the buffer is full. In this paper we show that the problem of detecting whether an MDP has a recurrent state is polynomial. We also show that the unichain classification problem for an MDP with a recurrent state is polynomial. In this paper, we call a state $i \in S$ *stopping* if $p(i|i,a) = 1$ for some $a \in A(i)$. The problem of detecting stopping states is polynomial and we also show that the unichain classification problem for an MDP with a stopping state is polynomial. We provide the algorithms and complexities to the corresponding problems.

Kallenberg [7] solved Some of classification problems in terms of graphs G^1 and G^2 whose arcs respectively represent that there are one-step transitions between two states for all actions and for some actions. According to the definition in [7], these graphs have no loops. We slightly modify the definition of graph G^2 by

*Eugene A. Feinberg is with the Department of Applied Math and Statistics, State University of New York at Stony Brook, NY 11794-3600, USA Eugene.Feinberg@sunysb.edu

†Fenghsu Yang is with the Department of Applied Math and Statistics, State University of New York at Stony Brook, Stony Brook, NY 11794, USA fyang@ic.sunysb.edu

978-1-4244-3124-3/08/$25.00 ©2008 IEEE

adding loops (i,i) if and only if $i \in S$ is a stopping state.

An MDP is called deterministic if $p(j|i,a) \in \{0,1\}$ for all $i,j \in S$ and for all $a \in A(i)$. For deterministic MDPs, the unichain classification problem is equivalent to the question whether the corresponding graphs G^2 have two node-disjoint cycles. This problem has been proved to be polynomial by McCuaig [9] and therefore the unichain classification problem for deterministic MDPs is polynomial.

In Section 2, we show that the unichain classification problem cannot be solved in terms of the graphs G^1 and G^2. In Section 3, we introduce the definitions of avoidable and reachable sets and provide the corresponding polynomial algorithms that finds the states from which a given set is avoidable and reachable. We also provide a polynomial algorithm that detects whether a state is recurrent and solves the unichain classification problem for an MDP with a recurrent state and a polynomial algorithm for detecting recurrent and stopping states and for the unichain classification problem with either recurrent or stopping states. Section 4 deals with detecting transient states in polynomial time and it discusses the implications of this capability for alleviating the complexity of the unichain classification problem.

2. Insufficiency of Graphs G^1 and G^2

Following Kallenberg [7], we define a directed graph G^1 to be a graph with the set of nodes S, no loops, and arcs (i,j), $i \neq j$, if and only if $\min\{p(j|i,a)|\, a \in A(i)\} > 0$. We also define a directed graph G^2 to be a graph with the set of nodes S such that: (i) an arc (i,j), $i \neq j$, belongs to G^2 if and only if $\max\{p(j|i,a)|\, a \in A(i)\} > 0$, and (ii) a loop (i,i) belongs to G^2 if and only if $p(i|i,a) = 1$ for some $a \in A(i)$. For a graph G, we also denote by G its incident matrix, i.e., $G(i,j) = 1$ if the arc (i,j) belongs to the graph and $G(i,j) = 0$ otherwise. The reason why we allow loops in graphs G^2 is because, as the following example shows, in the modified form the loops help us detect stopping states, while in the original form they do not.

Example 1 Let $S = \{1,2,3\}$, $A(1) = \{a,b\}$, $A(2) = A(3) = \{a\}$, $p(2|1,a) = p(3|1,b) = p(3|2,a) = p(2|3,a) = 1$. Observe that state 1 is not stopping. If we add an action c to $A(1)$ with $p(1|1,c) = 1$ then state 1 becomes stopping. The graph G^1 does not change. If we follow the definition in [7] that $G^2(i,i)$ is always 0, the graph G^2 does not change either. According to the above definition, $G^2(1,1)$ becomes equal to 1 and this detects that state 1 is stopping, if the action c is added. □

The following example shows that a unichain MDP and a multichain MDP have the same G^1 and G^2 graphs.

Example 2 Let $S = \{1,2,3,4\}$ and $A(i) = \{a,b,c\}$, $i = 1,2,3,4$. The first MDP is deterministic. Each action moves the process to a different state, and there are no stopping states. For example, in state 1, the action a moves the process to state 2, the action b moves the process to state 3, and the action c moves the process to state 4. This MDP is multichain. Indeed, if from state 1 (3) the process moves to state 2 (4) and from state 2 (4) the process moves to state 1 (3), then there are two recurrent classes $\{1,2\}$ and $\{3,4\}$.

The second MDP has the same state and action sets as the first one. All three actions define different transition probability vectors. For each action, the probability to stay in the same state is 0 and the probability to move to each of two remaining three states is 0.5. So, for state 1, we have $p(2|1,a) = p(3|1,a) = p(2|1,b) = p(4|1,b) = p(3|1,c) = p(4|1,c) = 0.5$. This MDP is unichain because the minimal possible number of states in a recurrent class is 3 and under all policies there are no absorbing states.

For the both graphs, we have that $G^1 = 0$ and $G^2(i,j) = 1$ if and only if $i \neq j$. □

In the following example, we provide two MDPs with identical corresponding graphs G^1 and G^2 such that one of these MDPs has no recurrent states and another one has a recurrent state. Therefore, the information provided by graphs G^1 and G^2 is insufficient to detect whether a state is recurrent.

Example 3 Consider two MDPs with $S = \{1,2,3,4\}$ and $A(i) = \{a,b,c\}$, $i = 1,2,3$, and $A(4) = \{a\}$. For both MDPs $p(1|4,a) = 1$. In states 1,2, and 3 the first MDP has the same transition probabilities as the first MDP in Example 2 and the second MDP has the same transition probabilities as the second MDP in Example 2. Since transition probabilities are the same at state 4, the corresponding graphs G^1 and G^2 coincide for these MDPs.

The first MDP has no recurrent states. Indeed, if we select actions in states 1,2, and 3 that move the process to state 4 then states 2 and 3 are transient and $\{1,4\}$ is a recurrent class. If we select in states 1 and 2 the actions that move the process to state 3 and in state 3 we select the action that moves the process to state 2 then 1 and 4 are transient states and $\{2,3\}$ is a recurrent class.

For the second MDP, state 1 is always recurrent. Indeed, for any deterministic policy any recurrent class contains at least three states. However, the process always moves from state 4 to state 1. Therefore, the set

$\{2,3,4\}$ cannot be a recurrent class for any deterministic policy. □

In the following example, we provide two MDPs with identical corresponding graphs G^1 and G^2 such that one of these MDPs has no transient states and another one has transient states. Therefore, the information provided by graphs G^1 and G^2 is insufficient to detect whether a state is transient.

Example 4 Consider two MDPs with $S = \{1,2,3,4\}$ and $A(i) = \{a,b,c\}$, $i = 1,2$, and $A(3) = A(4) = \{a\}$. The first MDP is deterministic. From states 1 and 2 it is possible to move to any other state. States 3 and 4 are absorbing. So, for the first MDP $p(2|1,a) = p(3|1,b) = p(4|1,c) = p(1|2,a) = p(3|2,b) = p(4|2,c) = p(3|3,a) = p(4|4,a) = 1$. Consider a policy that always selects the action a. Then the Markov chain has three recurrent classes: $\{1,2\}$, $\{3\}$, and $\{4\}$. So, this MDP has no transient state.

The second MDP has the same state and action sets as the first one with $p(2|1,a) = p(3|1,a) = p(2|1,b) = p(4|1,b) = p(3|1,c) = p(4|1,c) = p(1|2,a) = p(3|2,a) = p(1|2,b) = p(4|2,b) = p(3|2,c) = p(4|2,c) = 0.5$ and $p(3|3,a) = p(4|4,a) = 1$. This MDP has two transient states 1 and 2. In the both cases $G^1 = 0$. In the both cases: (a) $G^2(i,j) = 1$ when $i = 1,2$ and $j \neq i$, (b) $G^2(3,3) = G^2(4,4) = 1$, and (c) $G^2(i,j) = 0$ for other i and j. □

3. Definitions and Results

In this section, we define the avoidable and reachable sets and provide the polynomial algorithms that find the states from which a given set is avoidable and reachable respectively. Then we use the concepts of avoidable and reachable sets to detect whether a state is recurrent and solves the unichain classification problem for an MDP with a recurrent state. We will also show our main result of the polynomial algorithm for detecting recurrent and stopping states and for the unichain classification problem with either recurrent or stopping states.

3.1. Avoidable Set

Definition 1 *Let $i \in S$ and $Y \subset S$. The set Y is called avoidable from i if there exists a deterministic policy φ such that $P_i^{\varphi}(x_t \in Y) = 0$ for all $t = 0,1,\ldots$.*

A subset $Z \subseteq S$ is called *closed* under a deterministic policy φ if $p(i|j,\varphi(j)) = 0$ for any $j \in Z$ and for any $i \in S \setminus Z$. It is clear that a set Y is avoidable from i if and only if there exists $Z \subseteq S \setminus Y$ such that: (i) Z is closed under some deterministic policy, and (ii) $i \in Z$.

For $Y \subseteq S$ we let $Z^A(Y)$ be the set of $i \in S$ from which Y is avoidable. The following algorithm finds the set $Z^A(Y)$ for $Y \subseteq S$. Its convergence is based on the necessary and sufficient condition formulated in the previous paragraph.

Algorithm 1 *Finding $Z^A(Y)$ for a given $Y \subseteq S$.*

1. Set $Z := Y$, $\tilde{Z} := Y$.
2. Do while $\tilde{Z} \neq \emptyset$: for $j \in S \setminus Z$ set
$$A(j) := A(j) \setminus \{a \in A(j) | \sum_{l \in \tilde{Z}} p(l|j,a) > 0\}, \quad (1)$$
set $\tilde{Z} := \{j \in S \setminus Z : A(j) = \emptyset\}$, and set $Z := Z \cup \tilde{Z}$; end do.
3. Set $Z^A(Y) := S \setminus Z$. Stop.

The complexity of Algorithm 1 is $O(A \cdot N)$. Indeed, let $\tilde{Z}_t$, $t = 0,1,\ldots,m$, be the set $\tilde{Z}$ at the t^{th} iteration of Step 2, where $\tilde{Z}_0 = Y$, $\tilde{Z}_m = \emptyset$, and $m \leq N - |Y| + 1$, where $|E|$ denotes the number of elements in the finite set E. Observe that $\tilde{Z}_t \cap \tilde{Z}_s = \emptyset$ for $t \neq s$ and $\cup_{t=0}^{m-1} \tilde{Z}_t \subseteq S \setminus Y$. The complexity of computations in (1) at t^{th} iteration is $O(A \cdot |\tilde{Z}_t|)$, $t = 0,\ldots,m-1$. This implies that the complexity of Algorithm 1 is $O(A \cdot N)$.

Note that there are some similarities between the definition of an avoidable set above and the node that should be avoided in the Optimal Node Visitation (ONV) problem in stochastic graphs studied by Bountourelis and Reveliotis [3]. In particular, Algorithm 1 uses the same node elimination procedure as the independently formulated algorithm [3, Figure 3] for the reduction of the ONV problem.

3.2. Reachable Set

Definition 2 *Let $i \in S$, $Y \subset S$, and $i \notin Y$. The set Y is called reachable from i if there exists a deterministic policy φ such that $P_i^{\varphi}(x_t \in Y) > 0$ for some $t = 1,2,\ldots$.*

Note that the definition of a reachable set is slightly different than the standard definition of an accessible set since the former requires $i \notin Y$ and only considers $t > 0$. For $Y \subset S$, we denote by $Z^R(Y)$ the set of $i \in S$ from which Y is reachable. Finding $Z^R(Y)$ is equivalent to finding all the states from $S \setminus Y$ from which there is a path to Y in the graph G^2. The following algorithm finds the set $Z^R(Y)$ based on this concept.

Algorithm 2 *Finding $Z^R(Y)$ for a given $Y \subseteq S$.*

1. Construct the graph G^2. If Y is a singleton, let $Y = \{y\}$. If Y is not a singleton, reduce the set of nodes S by replacing the set Y with a single node $y \in Y$. Set $S^* := \{y\} \cup (S \setminus Y)$.

2. For all $i \in S \setminus Y$ set

$$G^2(i,y) := \begin{cases} 1 & \text{if } G^2(i,l) > 0 \text{ for some } l \in Y; \\ 0 & \text{otherwise;} \end{cases}$$

and reverse all the arcs in the reduced graph G^2.

3. For the starting node y, apply the breadth-first search algorithm [1, p.73-76]. $Z^R(Y)$ is the set of nodes, except y, in the breadth-first search tree.

The complexity of constructing the graph G^2 is $O(A \cdot N)$; see Kallenberg [7]. The complexities of Steps 2 is $O(N^2) \leq O(A \cdot N)$. The complexity of the breadth-first search algorithm is $O(N^2)$ [1, p.73-76]. Thus, the complexity of Algorithm 2 is $O(A \cdot N)$.

3.3. Finding Recurrent States

In this section, we formulate a polynomial algorithm that detects whether a particular state i is recurrent. Moreover, if the state is recurrent, the polynomial algorithm also detects whether the MDP is unichain. Example 3 indicates that finding a recurrent state in an MDP cannot be done by using only matrices G^1 and G^2.

If Y contains only one state, $Y = \{i\}$, we shall write $Z^A(i)$ and $Z^R(i)$ instead of $Z^A(Y)$ and $Z^R(Y)$ respectively. We apply Algorithm 1 to $Y = \{i\}$ and find the set $Z^A(i)$. If $Z^A(i) = \emptyset$, it is obvious that i is recurrent and the MDP is unichain. If $Z^A(i) \neq \emptyset$, we apply Algorithm 2 to $Y = Z^A(i)$ and find the set $Z^R(Z^A(i))$. If $i \in Z^R(Z^A(i))$ then $Z^A(i)$ is reachable from i and i is avoidable from any $j \in Z^A(i)$. Therefore, i is not recurrent and we do not know whether the MDP is unichain or multichain. On the other hand, if $i \notin Z^R(Z^A(i))$, then, starting from i, the process will never reach $Z^A(i)$ and will travel only through the states from which i is not avoidable. In this case, we know i is recurrent and the MDP is multichain because there is a subset of $Z^A(i)$ which forms a recurrent class for a Markov chain defined by some deterministic policy. The following algorithm detects whether a state i is recurrent. If the state is recurrent, the algorithm also detects whether an MDP is unichain.

Algorithm 3 *Detecting whether a state i is recurrent and, if i is recurrent, whether the MDP is unichain.*

1. Apply Algorithm 1 to find $Z^A(i)$. If $Z^A(i) = \emptyset$ then the state i is recurrent and the MDP is unichain, and stop.

2. Apply Algorithm 2 to find $Z^R(Z^A(i))$. If $i \in Z^R(Z^A(i))$ then the state i is not recurrent. Else, the state i is recurrent and the MDP is multichain. Stop.

The complexity of Algorithm 3 is $O(A \cdot N)$ because Algorithms 1 and 2 have this complexity. We may have to apply Algorithm 3 to each state in order to detect if there exists a recurrent state in an MDP. This procedure leads to construct the set of recurrent states and its complexity is $O(A \cdot N^2)$. Repeating Algorithm 3 at most N times until a recurrent state is found also leads to the solution of the unichain classification problem for an MDP with a recurrent state. The complexity of this algorithm is $O(A \cdot N^2)$ too. In the following section we provide an algorithm for solving a unichain classification problem for an MDP with either a recurrent or stopping state.

3.4. Polynomial Algorithm to Detect if an MDP with either a Recurrent or Stopping State is Unichain

If a state i is either recurrent or stopping then the MDP is unichain if and only if under all deterministic policies there is no recurrent class that does not contain i. Let a state i be either stopping or recurrent. If $Z^A(i) = \emptyset$ then the state i is unavoidable from all other states. In this case, under any deterministic policy any recurrent class contains i. Therefore, the MDP is unichain. On the other hand, if $Z^A(i) \neq \emptyset$ then under some deterministic policy the corresponding Markov chain contains a recurrent class that does not contain the state i. Obviously, the MDP is multichain.

If an MDP contains more than one stopping state, it is multichain obviously. Even though there may be two or more recurrent states in the MDP, we only need

to apply Algorithm 1 to one recurrent or stopping state in order to know whether the MDP is unichain. Thus, we can formulate the following algorithm.

Algorithm 4 *Polynomial Algorithm to Detect whether an MDP has a stopping or recurrent state and, if it does, whether an MDP is unichain.*

1. For $i = 1,\dots,N$ and for $a \in A(i)$ check the condition $p(i,i) = 1$ until two stopping states are found.
2. If two stopping states are found, the MDP is multichain and stop.
3. If one stopping state i is found then apply Algorithm 1 with $Y = \{i\}$ and
   (a) if $Z^A(i) \neq \emptyset$, the MDP is multichain and stop;
   (b) if $Z^A(i) = \emptyset$, the MDP is unichain and stop.
4. For $i = 1,\dots,N$ apply Algorithm 3 as long as a recurrent state is not found. Stop after a recurrent state is found and the MDP is classified.
5. Conclude that the MDP contains neither stopping nor recurrent states and stop.

The complexity of Algorithm 4 is $O(A \cdot N^2)$ since it requires running Algorithm 3 at most N times.

4. Finding Transient States and Remarks

Let T be the set of transient states. This set can be found by apply Bather's decomposition algorithm [2] This algorithm is formalized in [7, Algorithm 7] and its complexity is $O(A \cdot N^2)$ [7]. In terms of [7, Algorithm 7], the set of transient states T is the union of the sets $T_1,\dots,T_m$ computed by that algorithm.

After the set of transient states T is found, we can delete T from the state space S and reduce the action sets $A(j)$, $j \in S \setminus T$, to

$$A(j) := A(j) \setminus \{a \in A(j) : \sum_{i \in T} p(i|j,a) > 0\}. \qquad (2)$$

Any deterministic policy φ in the original MDP defines a deterministic policy in the reduced MDP as a function on $S \setminus T$. Since the states in T are always transient in the original MDP, the recurrent classes for these two Markov chains coincide. Thus, it is easy to see that the original MDP is unichain if and only if the reduced MDP is unichain. Thus, if $T \neq \emptyset$, by removing the set T and reducing the actions, we can reduce the unichan classification problem to a smaller problem.

An MDP is called *communicating* if for each two states $i, j \in S$ there exists a deterministic policy φ, which may depend on i and j, such that j accessible from i in the Markov chain defined by φ. An MDP is called *weakly communicating* if, after the set T is deleted and the action sets in $E := S \setminus T$ are reduced following (2), the MDP with the state space E is communicating. If an MDP is not weakly communicating, it is multichain. This follows from Bather's [2] decomposition.

Algorithm 4 in [7] detects whether an MDP is weakly communicating and its complexity is $O(A \cdot N^2)$. If an MDP is weakly communicating, it can be reduced in polynomial $O(A \cdot N^2)$ time to a communicating MDP; see (2). Thus, the unichain classification problem for a weakly communicating MDP can be reduced in polynomial ($O(A \cdot N^2)$) time to an *NP*-hard unichain classification problem for a communicating MDP.

Algorithm 4 solves the unichain classification problem for MDPs with recurrent and stopping states. Algorithm 5 in Kallenberg [7] also solves the unichain classification problem for some MDPs. Both algorithms have complexity $O(A \cdot N^2)$. [7, Algorithm 5] finds strongly connected components (maximal connected subsets) of the graph G^1. Then it compresses G^1 by replacing each strongly connected component in G^1 with a single node. In the compressed graph, there exists an arc (i^*, j^*) if in the strongly connected component corresponding to i^* there is a state i such that $\sum_{j \in X} p(j|i,a) > 0$ for all $a \in A(i)$, where X is the strongly connected component compressed into j^*. Then [7, Algorithm 5] conducts additional compressions by merging nodes i^* with the nodes j^* if the arc (i^*, j^*) exists and i^* is with outgoing rank 1. These procedures are repeated recursively until the graph cannot be compressed anymore. Let $(G^1)^+$ be the graph that is eventually obtained and cannot be compressed and k^+ be the number of strongly connected components in $(G^1)^+$. If $k^+ = 1$ then the MDP is unichain, if $k^+ = 2$ then the MDP is multiichain, and if $k^+ = 3$ then the MDP is either unichain or multichain; [7, Theorem 3.6].

At the end of this section, we give two examples to show that Algorithm 4 in this paper and [7, Algorithm 5] solve different classes of problems. Of course, these two classes overlap. Algorithm 4 always classifies an MDP with a recurrent state. It is clear that, if $k^+ = 1$, [7, Algorithm 5] compresses the graph around a recurrent state. Thus, if [7, Algorithm 5] detects that the MDP is unichain, this MDP has a recurrent state. Example 5 provides a unichain MDP with a recurrent states and this MDP cannot be classified by [7, Algorithm 5]. Example 6 shows that [7, Algorithm 5] can classify some MDPs without recurrent states.

Example 5 Consider an MDP with the same state and action sets as in the second MDP in Example 3. In states 1,2, and 3, the transition probabilities are the same as in the second MDP in Example 3. In addition, $p(1|4,a) = p(2|4,a) = p(3|4,a) = \frac{1}{3}$. In this MDP, states 1,2, and 3 are recurrent. For this MDP, $G^1(4,j) = 1$, $j = 1,2,3$, and $G^1(i,j) = 0$ in all other cases. This graph cannot be compressed and therefore $k^+ = 4$. □

Example 6 Let $S = \{1,2,3,4\}$, $A(1) = A(3) = \{a\}$, and $A(2) = A(4) = \{a,b\}$. In addition, $p(2|1,a) = p(1|2,a) = p(4|3,a) = p(3|4,a) = 1$ and $p(1|2,b) = p(3|2,b) = p(3|4,b) = p(1|4,b) = \frac{1}{2}$. This MDP has no recurrent states. The graph G^1 has two strongly connected components $\{1,2\}$ and $\{3,4\}$ and they contract to a graph consisting of two isolated nodes. Thus $k^+ = 2$ and [7, Algorithm 5] detects that this MDP is multichain. □

5. ACKNOWLEDGMENTS

This research was partially supported by NSF grant DMI-0600538. The authors thank Spyros Reveliotis for valuable comments and sending their working paper [3]. The first author thanks Noga Alon for providing the reference to [9].

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows*, Prentice Hall, New Jersey, 1993.

[2] J. Bather, Optimal decision procedures for finite Markov chains. Part III. *Advances in Applied Probability* **5**(1973), pp. 541–553.

[3] T. Bountourelis, S. Reveliotis, Optimal Node Visitation in Stochastic Digraphs. Preprint. School of Industrial & Systems Engineering. Georgia Institute of Technology, 2007.

[4] C. Derman, *Finite State Markov Decision Processes*, Academic Press, New York, 1970.

[5] E.B. Dynkin, A.A. Yushkevich, *Controlled Markov Processes*, Springer-Verlag, New York, 1979.

[6] R.A. Howard, *Dynamic Programming and Markov Processes*, MIT Press, Cambridge, MA, 1960.

[7] L.C.M. Kallenberg, Classification problems in MDPs, in: Z. How, J.A. Filar and A. Chen (Eds.), *Markov Processes and Controlled Markov Chains*, Kluwer, Boston, 2002, pp. 151-165.

[8] L.C.M. Kallenberg, Finite state and action MDPs, in: E.A. Feinberg and A. Shwartz (Eds.), *Handbook of Markov Decision Processes*, Kluwer, Boston, 2002, pp. 21–87.

[9] W. McCuaig, Intercyclic digraphs, in: N. Robertson and P. Seymour (Eds.), Graph Structure Theory, *Contemporary Mathathematics*, vol. 147, Amer. Math. Soc., Providence, RI, 1993, pp. 203–245.

[10] M.L. Puterman, *Markov Decision Processes*, Wiley, New York, 1994.

[11] J.N. Tsitsiklis, *NP*-hardness of checking the unichain condition in average cost MDPs. *Oper. Res. Lett.* **35**(2007), pp. 319–323.