International Journal of Advanced Biotechnology and Research (IJBR)
ISSN 0976-2612, Online ISSN 2278–599X,
Vol-7, Special Issue-Number5-July, 2016, pp59-63
http://www.bipublication.com

Research Article

Accelerating of Modified Policy Iteration in Probabilistic Model Checking

Mohammadsadegh Mohagheghi

Departement of Computer Science,

Vali-e-asrRafsanjan University

mohagheghi@vru.ac.ir

ABSTRACT

Markov Decision Processes (MDPs) are used to model both non-deterministic and probabilistic systems. Probabilistic model checking is an approach for verifying quantitative properties of probabilistic systems that are modeled by MDPs. Value and Policy Iteration and modified version of them are well-known approaches for computing a wide range of probabilistic properties. This paper tries to improve the performance of modified policy iteration. Our approach is to use some information of the related model for approximating a good value for the number of iterations for each policy and improving this approximation in next modifications of policies.

Keywords: Software Verification, Formal methods, Model checking, Probabilistic Systems.

1. INTRODUCTION

Model checking is an automated formal verification approach which is used to guarantee the correctness of computer systems. Because of some stochastic behaviors of these systems, probabilistic model checking can be used to analyze quantitative properties. Markov Decision Processes are a well-known formalism for modeling systems with both probabilistic and nondeterministic behaviors. Main problems in the verification of quantitative properties of MDPs can be reduced to probabilistic reachability problems, i.e., the minimum or maximum probability that a set of paths through the MDP finally reach one of goal states.

There are some algorithms for computing these probabilities. Linear programming is an approach for computing the exact values of these probabilities. However, its limited scalability forlarge models are a main drawback of this approach. Numerical methods like value and policy iteration are used in practice to approximate the values of probabilities. Value iteration, updates the value of every state of the model in every iteration. It updates the value of states until reaching the stopping condition. In policy iteration, the algorithm consists of several policy selections and for every selection, it computes the value of states like value iteration method [1].

There are some other methods like Gauss-Seidel and modified policy iteration that use the ideas of value and policy iteration methods. The performance of these algorithms depends on many factors and some improved methods have been proposed to accelerate these iterative algorithms. Strongly-connected component (SCC) and learning based methods can improve the mentioned iterative methods by avoiding redundant or useless updates of states and by updating states in the appropriate order[2,3]. Symmetric reduction is another method that is useful for systems with a set of non-trivial, but interchangeable components [4].

Experimental results show that the performance of most improved methods is not more than standard iterative ones in the case of dense and also asymmetric models. In this case, policy iteration and the modified version of it, obtain

better performance for many samples. Hence, in this paper we concentrate in improving the performance of modified policy iteration.

One factor that affects the performance of modified policy iteration is the number of iterations after each modification of policies. While in the standard version of this algorithm, there area fixed number of iterations for each policy (100 numbers in PRISM model checker [6]), we use a method that defines the number of iterations for every modification of policies, using some information of the model and previous policies.

The contribution of this work is to define a dynamic stopping criterion for each policy in modified policy iteration algorithm. This new criterion accelerates modified policy iteration up to40%. For experimental results, we use both sparse and dense MDPs.

The rest of this paper is organized as follows: Section 2 presents some preliminaries about probabilistic verification of MDPs. Section 3 describes our idea for accelerating modified policy iteration algorithm. Experimental results are presented in section 4and section 5 concludes the short-paper.

2. PRELIMINARIES

For a countable set S, a discrete probability distribution on S is defined as afunction $P: S \to [0,1]$ such that $\sum_{s \in S} P(s) = 1$.In this section weprovide an overview of MDP [7].

Definition 1 Markov Decision Processes (MDPs)

Given finite sets S, Act and L, an MDP is defined as a tuple $M = (S, Act, P, AP, L)$ where S is a countable set of states; Act is afinite set of actions; $P: S \times Act \times S \to [0,1]$is the transitionprobability function such that for each state $s \in S$and each action $\alpha \in Act$: $\sum_{s' \in S} P(s, \alpha, s') \in \{0,1\}$; AP is a non-empty set ofatomic propositions;and $L: S \to 2^{AP}$is a labeling function. Inthis definition,transition is a triple (s, α, s') for which $s, s' \in S$ and $\alpha \in Act$and $P(s, \alpha, s') > 0$.

The size of S is defined as the total number of states and isshown by |S|. We use $P(s, \alpha)$as the set of all states $s' \in S$where $P(s, \alpha, s') > 0$. An action $\alpha$is enabled in the state s if and only if $\sum_{s' \in S} P(s, \alpha, s') = 1$.For each state $s \in S$, we use $|Act(s)|$for thenumber of enabled actions of s, and $|Act|$ for the total number ofactions of the model and $|P|$ for the total number of transitions. The size of M is defined as $|M| = |S| + |P|$.

A path in an MDP, is a non-empty (finite or infinite)sequence of states and actions of the form $\omega = s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} s_2 \xrightarrow{\alpha_2} \dots$ where $P(s_i, \alpha, s_{i+1}) > 0$ for each $i \geq 0$. We use $\omega(i)$to denote the(i+1)th state in the path ω. $Path_s$denotes the set of all pathsstarting in state s and $FPath_s$the set of all finite paths.

In a sample execution of an MDP, there are two steps to determine the successor of a state s.First an action $\alpha \in Act(s)$ischosen non-deterministically; Then the next state $s' \in P(s, \alpha)$ischosen randomly according to the probability distribution P. To resolve nondeterministic choices in an MDP $M = (S, Act, P, AP, L)$, we require the notion of policy (sometimes called scheduler oradversary). A policy is a function $\sigma: FPath \to Act$which given afinite path, selects an enabled action in each state, based on thehistory of choices made so far (or simply the last state in memory-less policies). The set of all policies of $M$is Adv_M. A policy σrestricts the behavior of the MDP to a set of paths $Paths_s^\sigma \subseteq Paths_s$and a probability space $Prob_s^\sigma$is defined over these paths.Considering $G \subseteq S$ as a set of goal states of the MDP, we use $P_s^{max}(G)$(respectively $P_s^{min}(G)$ as the maximum (respectivelyminimum) probability of reaching G from the state s. Formally $P_s^{max}(G)$ and $P_s^{min}(G)$ are defined as:

$$P_{in}^{max}(G) = \sup_{\sigma \in Adv_M} P_{in}^{\sigma}(G) \text{ and}$$
$$P_{in}^{min}(G) = \inf_{\sigma \in Adv_M} P_{in}^{\sigma}(G) \quad (1)$$

Where

$P_s^\sigma(G) - Probs_s^\sigma(\{\omega \in Paths_s^\sigma | \exists i. \omega(i) \in G\})$

. More details of this definition can be found in [2,7].

2-1 Value Iteration (VI) method

This method approximates the values of $P_s^{max}(G)$ for every state s iteratively, until the values of all states converge with respect to agiven level of precision $\varepsilon > 0$. Algorithm 1 gives the idea of the valueiterationmethod. Firstly, the algorithm uses some pre-computation to compute those states for which the maximum probability of reaching goal states is equal to one[2].After this pre-computation, the algorithm computes the values of x_s iteratively until satisfying termination criterion.

Algorithm 1 Value Iteration for computing $P^{max}(G)$

1: foreach $s \in S$ do $x_s := \begin{cases} 1 \text{ if } s \in G \\ 0 \text{ otherwise} \end{cases}$
2: do
3: foreach $s \in S \backslash G$ do
4: $x'_s := \max_{\alpha \in Act(s)} \sum_{s' \in S} P(s,\alpha,s').x_{s'}$;
5: end
6: $\delta = \max_{s \in S}(x'_s - x_s)$;
7: foreach $s \in S \backslash G$ do $x_s := x'_s$ end for
8: while $\delta > \epsilon$;
9: return $(x'_s)_{s \in S}$

2-2 Policy Iteration (PI) method

This algorithm starts with an arbitrary policy σ and computes theprobability of $P_s^{\sigma}(G)$ for every state s.Then; it repeatedly improves the policy by computing the corresponding probabilities and selecting the actions taken so that the reachability probabilities are increased. Computing the probabilities $P_s^{\sigma}(G)$ for each policy σ is done by computing reachability probabilities for the corresponding Discrete Time Markov Chain.

This part of computations can be done by using iterative methods for computing reachability probabilities in DTMCs [7]. Algorithm 2gives the idea of policy iteration.

Algorithm 2 Policy Iteration computing $P^{max}(G)$

1: select an arbitrary policy σ;
2: do
3: compute $p_s := P_s^{\sigma}(G)$ for all $s \in S$
4: foreach $s \in S \backslash G$ do
5: $\sigma(s) := \text{argmax}_{\alpha \in Act(s)} \sum_{s' \in S} P(s,\alpha,s').x_{s'}$;
6: end for
7: while σ has changed;
8: return $(p_s)s \in S$

2-3 Modified Policy Iteration

One drawback of the PI method is that the number of iterations(and computations) after improving each policy is high, whileevery policy (except the last one) is an approximation of the bestone and some non-optimal action selections can affect the speedof convergence for the method. In fact, these approximations areuseful only for a limited number of iterations and can be modifiedafter a predefined number of iterations. It is the idea of modifiedpolicy iteration method and is summarized in Algorithm 3. Thismethod has been proposed to solve the minimum expected costproblem in the field of reinforcement learning [5] and also is implemented in the PRISM model checker [6].

There are two main differences between Algorithm 3 andAlgorithm 2:

- In Algorithm 3, the number of iterations for each policy(lines 4-11) is limited to a fixed value.
- The termination criterion of Algorithm 3 is like value iteration (algorithm 1). It terminates when the maximum changes of value of states, using all enabled actions, drops below a specified threshold ϵ.

3- Accelerating Modified PI

In this section, we analyze the performance of modified PI and compare it, with the performance of VI and PI methods. Because the runtime of Algorithm 3 is a function of number of multiplications in lines 7 and 15, we try to reduce the total number of multiplications.

Algorithm 3 Modified Policy Iteration

1: select an arbitrary policy σ
2: do
3: $i := 1; \delta := 1; k := 100;$
4: while $i \leq k$ and $\delta > \epsilon$
5: $\delta := 0;$
6: foreach $s \in S \backslash G$ do
7: $x_{new} := \sum_{s' \in S} P(s, \alpha, s').x_{s'};$
8: $\delta := \max(\delta, x_{new} - x_s)$
9: $x_s := x_{new}$
10: end for
11: end while
12: $\delta := 0;$
13: foreach $s \in S \backslash G$ do
14: $\sigma(s) := \text{argmax}_{\alpha \in Act(s)} \sum_{s' \in S} P(s, \alpha, s').x_{s'};$
15: $x_{new} := \sum_{s' \in S} P(s, \alpha, s').x_{s'};$
16: $\delta := \max(\delta, x_{new} - x_s)$
17: $x_s := x_{new};$
18: end for
19: while $\delta > \epsilon$

Let ϑ be the average number of actions per state, i.e., $\vartheta = \frac{|Act|}{|S|}$ .Moreover, let Itr_{VI} and Itr_{PI} and Itr_{MPI} be the total number ofiterations in the value and policy and modified policy iterationmethods respectively and Itr_{mod} the number of policymodifications. As the result the total number of multiplicationsfor the VI method is equal to $Itr_{VI} \times |P|$ and for the PI is equal to

$$Itr_{mod} \times |P| + (Itr_{PI} - Itr_{mod}) \times \frac{|P|}{\vartheta} \quad (2).$$

Unfortunately, there is not any good way to estimate the values of Itr_{VI} and Itr_{PI} unless running the related methods. There is a comparison of the VI and PI and G-S methods for some case studies in [1].

3-1Analysis of the performance of modified policy iteration

The total number of multiplications in the modified policy iteration is approximately equal to $Itr_{mod} \times \left(\frac{k}{\vartheta}\right) \times |P|$ (3)

Again there isn't any direct method for calculating this value but, good values for k can accelerate modified PI. Small values for k results more policy modifications and generally betterapproximation for each policy. We consider two cases forclarifying: in the first, we set the value of to 100 and in thesecond, we set it to 9. After 202 overall iterations in the first case,we have only two policy modifications, while after 200 in thesecond, we have 20 policy modifications and it is more likely tohave a better policy in the second case. On the other hand theoverhead of policy modification in the second case is more, i.e.,by reducing the value of k, the value of Itr_{mod} becomes high andthe total number of multiplications for policy modificationsbecome more. The overhead of policy modification, in general,depends on the parameter ϑ and as a result, one can define thevalue of k as a function of ϑ.

3-2 Improving Modified PI

For improving the performance of modified PI, we should improve approximations of policies. In most of case studies, the first constructed policy is far from the optimal one but the policies incrementally converge to the optimal. It influences the value of k:If the accuracy of the constructed policy is low, the value of $k$should be low. It avoids useless updates and modifies policy aftersmall number of iterations. On the other hand if the constructedpolicy is more accurate, it can be used for more iteration.In the latter case, the value of k should be high. We use number ofactions that are changed in two consequent policies as a criterionfor the accuracy of the current policy and show it by k. Usingthis fact and experimental results, we propose the following relation for the value k:

$$k := \vartheta + \frac{|S|}{2 \times m + 2} \quad (4)$$

In fact it is a heuristic for the value of and we define this relation according to empirical results. When the selected actionof most states is changed in new policy, the value of m will behigh. In this case k is low and it causes sooner modification in thepolicy.

Small values for m mean good approximation for thecurrent policy and causes higher value for k and more iteration forthis policy reduces the overhead of policy modification.

4- Experimental results

Table 1 shows experimental results for some case studies: Consensus (N = 4 , K = 4) from the

PRISM benchmark suite, Client-Server (1 failure with N = 7) from [8] , SysAdmin8 andSysAdmin10 from [5], and M500 and M2000, two randomly generated MDP with 500 and 2000 states.

The first two, are sparse models and we also use a state ordering method for them.Our ordering method is based on a backward BFS that starts fromgoal states.

SysAdmin8, SysAdmin10, M500 and M2000 aredense models. The table shows number of multiplications (as the main operation of computations) for everymodel using Gauss-Seidel, modified PI with constant k (= 100)and with our method for dynamic value for k. For example number of multiplications in the consensus example is approximately 277×10^6.

Model	Size	θ	Gause-Seidel	Modified PI with k = 100	Modified PI whith dynamic k
Consensus (Initial ordering)	43136	2.7	277M	149.5M	147.8M
Consensus (Improved ordering)	43136	2.7	90M	47.8M	49.6M
Client-Server(Initial ordering)	130323	4	31.7M	41M	24.8M
Client-Server(Improved ordering)	130323	4	27.1M	25M	14.7M
SysAdmin8	256	8	79.1M	19.3M	11.6M
SysAdmin10	1024	10	563.M	173.8M	122.6M
M500	500	4	124M	45.4M	31.1M
M2000	2000	5	1971M	474M	428M

Table 1. Performance comparison for 8 case studies.

The table shows considerable improvement in the performance of modified PI for 5 cases. For the Consesus model, our approach doesn't accelerate modified PI. The main reason is that in this case, the total number of iterations is high (more than 10000).However; previous works like symmetric reduction out perform modified PI for this case study [4].

5-CONCLUSION AND FUTURE WORKS

In this short paper, we propose a heuristic for stopping criterionfor modified policy iteration method and compare the impact ofthis heuristic with other iterative methods. For future work, we tryto improve the proposed heuristic and also use the idea of actionelimination to remove useless actions and improve theperformance of modified policy iteration. We also plan to work onmore case studies and generalized our heuristic.

6- REFERENCES

1. Forejt, Vojtěch, et al. "Automated verification techniques forprobabilistic systems." Formal Methods for Eternal Networked Software Systems. Springer Berlin Heidelberg,2011. 53-113.
2. Kwiatkowska, Marta, David Parker, and Hongyang Qu."Incremental quantitative verification for Markov decisionprocesses." Dependable Systems & Networks (DSN), 2011IEEE/IFIP 41st International Conference on. IEEE, 2011.
3. Brázdil, Tomáš, et al. "Verification of Markov decisionprocesses using learning algorithms." Automated Technologyfor Verification and Analysis. Springer InternationalPublishing, 2014. 98-114.
4. Kwiatkowska, Marta, Gethin Norman, and David Parker."Symmetry reduction for probabilistic model checking." Computer Aided Verification. Springer BerlinHeidelberg, 2006.
5. Wingate, David, and Kevin D. Seppi. "Prioritization methodsfor accelerating MDP solvers." Journal of Machine LearningResearch. 2005.
6. http://www.prismmodelchecker.org.
7. C. Baier, and J.-P. Katoen, Principles of model checking:MIT press Cambridge, 2008.
8. L. Feng, "On Learning Assumptions for CompositionalVerification of Probabilistic Systems," University of Oxford,2013.