UDK 004

Policy Based Interval Iteration for Probabilistic Model Checking

Mohagheghi M. (Vali-e-Asr University of Rafsanjan, Rafsanjan, Iran)

Khademi A. (Vali-e-Asr University of Rafsanjan, Rafsanjan, Iran)

Reachability probabilities and expected rewards are two important classes of properties that are used in probabilistic model checking. Iterative numerical methods are applied to compute the underlying properties. To guarantee soundness of the computed results, the interval iteration method is used. This method utilizes two vectors as the upper-bound and lower-bound of values and uses the standard value iteration method to update the values of these vectors. In this paper, we use a combination of value iteration and policy iteration to update these values. We use policy iteration to update the values of the lower bound vector. For the upper-bound vector, we use a modified version of value iteration that marks useless actions to disregard them for the remainder of the computations. Our proposed approach brings an opportunity to apply some advanced techniques to reduce the running time of the computations for the interval iteration method. We consider a set of standard case studies and the experimental results show that in most cases, our proposed technique reduces the running time of computations.

Keywords: Probabilistic model checking, Sound iterative methods, Policy iteration

1. Introduction

Value iteration (VI) and policy iteration (PI) are two well-known iterative numerical methods that are used to approximate the required values in the analysis of Markov decision processes. In reinforcement learning, VI and PI are used to compute the optimal expected rewards (or costs) for an intelligent agent. In formal verification, these two methods are also used to compute the optimal reachability probabilities: the minimal or maximal probability of reaching a goal state. In both cases, linear programming (LP) can be used to compute exact values. Because of the scalability of this technique, LP can not be used for large models. VI and PI are used instead to approximate the required values. These techniques compute iteratively until satisfying a stopping criterion[1, 5]. In reinforcement learning, a discount factor is considered to guarantee the convergence of computations. The iterative computations terminate when the maximal difference between two consecutive iterations drop below a threshold [5, 6]. In formal verification, where the optimal probabilities or expected rewards until reaching a goal state is needed, some pre-computations are applied to disregard those states for which, the accumulated expected rewards converge to infinite. For the remaining states, VI or PI can be applied. Both approaches iteratively update state-values until reaching the case where the difference between values drops the threshold. Although in most cases, this stopping criterion is sufficient to verify the correctness of a given specified property, there is not any guarantee for the precision of the computed values. There are some cases in probabilistic model checking where VI or PI terminate and the reported reachability probability (or expected reward) is far away from the exact value [2, 9]. To overcome this deficiency, the interval iteration method has been proposed as an extension to the standard value iteration and is implemented in most model checker tools. In interval iteration (II), two vectors of values are considered as an upper-bound and lower-bound of the state-values. These vectors are updated iteratively until reaching a case where the maximal difference of state values between these two vectors drop the threshold. In such case, it is guaranteed that the exact value is between the two ones and maximal error is controlled [4, 9, 15].

The running time of the standard iterative methods for probabilistic model checking is an important challenge that affects their application on large models. While the number of states of a model can grow exponentially in the number of components, that causes the state explosion problem, a modern model checker should control the total number of numerical computations. Several approaches have been proposed to accelerate the VI and PI methods. The results of [13] demonstrate that the modified and improved versions of PI can outperform VI for most standard case study models. For the interval iteration method, using two vectors of values brings more computations and the running time is more than the VI and PI methods. For this method, it is important to utilize improved techniques to reduce the overall running time. In most previous works, VI is mainly used to update the values of these vectors, i.e., in each iteration, VI is used to update the upper-bound and lower-bound values of each state. Appropriate state ordering for VI has been used as an approach to accelerate the interval iteration method [8]. This approach is applied in [2, 15] to reduce the overall running time of the computations. To the best of our knowledge, no other work has considered more advanced techniques to accelerate the iterative computations for the II method. In this paper, we focus on the problem of accelerating II and select PI as the standard method for updating the state-values of this method. We utilize several accelerating methods for PI to improve the performance of II. The main benefit of these accelerating methods is that in most iterations, they disregard less important computations and focus on more important ones.

1.1. Related work

Value iteration and policy iterations have been used as the standard iterative methods in probabilistic model checking [1, 3, 5]. However, the soundness of these methods was studied for the first time in [7] where a counterexample is proposed that shows VI may terminate and return a result that is far away from the exact value. To cover this problem and guarantee the soundness of the computed values, the interval iteration method is proposed in [7] for the extremal reachability probabilities. The extension of II for the extremal expected rewards is proposed in [2] where some approaches are suggested for computing the upper-bound vector of values. In some cases, the computed upper-bounds are far away from true values that causes a large number of iterations in the computations of the II method. To reduce the overall running times and start from better upper-bounds, the sound value iteration[15] and optimistic value iteration[10] methods are proposed for computing the minimal and maximal expected rewards. The possibility of applying policy iteration for the interval iteration method is studied in [9]. It has only considered the standard PI method and do not studied the impact of modified policy iteration or other improved technique on the performance of computations.

2. Preliminaries

In this section, we review some important concepts of probabilistic model checking. More details are available in [2, 3, 8]. For a finite set S and vectors $\underline{x} = (x_s)_{s\in S} \in \mathbb{R}^{|S|}$ and $\underline{y} = (y_s)_{s\in S} \in \mathbb{R}^{|S|}$, we write $\underline{x} \leq \underline{y}$ if $x_s \leq y_s$ for all $s \in S$ and we write $\underline{x} < \underline{y}$ if $x_s < y_s$ for all $s \in S$.

2.1. Markov Decision Process

Definition 1. A Markov Decision Process (MDP) is a tuple $M = (S, s_0, Act, P, R)$ where:

- S is a finite set of states,
- $s_0 \in S$ is the initial state,
- Act is a finite set of actions.
- $P : S \times Act \times S \to [0, 1]$ is a probabilistic transition function such that for each state s and the action $\alpha \in Act$ we have $\sum_{s'\in S} P(s, \alpha, s') \in \{0, 1\}$.

- $R : S \times Act \to \mathbb{R}$ is a reward function.

An action α is *enabled* in state s, if $\sum_{s' \in S} P(s, \alpha, s') = 1$. The operational semantics of MDP M is as follows. M is initiated with state s_0. Assume that M is in state s. First, one of the enabled actions of s is selected non-deterministically. According to the selected action α, the reward $R(s, \alpha)$ is collected by the system. Then due to probabilistic transition function P, one of the α-successor states of s (a state for which there is a transition from s via action α) is chosen. That is, s' is the next state with probability $P(s, \alpha, s')$. A discrete-time Markov chain (DTMC) is an MDP in which every next state is selected just probabilistically, i.e. it has exactly one enabled action [1].

The minimal (or maximal) probability of reaching one of the goal states is called extremal reachability probability. Extremal expected rewards are defined as the minimal (or maximal) expectation of accumulated rewards until reaching a goal state. For a state $s \in S$, we use $\mathbb{E}_s^{max}$ to denote the maximal expected reward. Some graph-based methods are exploited to detect the set of states that the extremal reachability probability is one [1, 3]. These sets are used to computing the maximal or minimal expected rewards [8].

A standard way to resolve non-deterministic choices in MDPs is to define and use policies. A deterministic policy (also called adversary) is function $\pi : S \to Act$ that maps each state of the MDP to one of its enabled actions. A policy is memory less where it decide the actions only based on the last visited state [8]. For unbounded reachability probabilities and expected rewards, the optimal action of each state $s \in S$ is determined based of the probability distribution of the outgoing transitions and the computed values of the successor states and it is not important which states have been visited before reaching s. Hence, for these classes of properties, that are the topic of the current work, it is sufficient to consider deterministic and memory-less policies [1, 3].

Iterative numerical methods are used to approximate the values of the expected rewards. Value iteration, as a well known method uses a vector x^k to store the approximated values of the maximal expected rewards in iteration k. The values of x_s^k determines the approximated value of $\mathbb{E}_s^{max}$ after k iterations. In each iteration k the value of x^k is computed according to the computed value of x^{k-1} from the previous iteration. Theoretically, value iteration can finally converge to the exact expected values, that is, $\lim_{k \to \infty} x_s^k = \mathbb{E}_s^{max}$; but practically, a convergence criterion is exploited to terminate the iterations. To do so, the maximum difference of computed values between two consecutive iterations are compared with a threshold ϵ in which the value iteration method terminates when $max_{s \in S^1_{min}}(x^k_s - x^{k-1}_s) < \epsilon$ [8]. In the literature, several methods are proposed to accelerate the standard iterative methods using state prioritizing approaches [8, 13]. Policy iteration (PI) is another option for approximating the extremal reachability probabilities or expected reward[5]. This method follows a set of rounds. In each round, PI considers a policy to select one action for each state. Based on the selected actions a DTMC is induced and a set of iterations is applied to update the state values. For the first round, the policy is defined by randomly selecting the actions of each state. For the other rounds, policies are defined by considering the best action of each state by considering the computed values of the previous round. The computations continue until reaching a stable policy, where the last two policies are the same [9].

2.2. Interval Iteration for Expected Accumulated Rewards

Value iteration with the standard termination criterion can not guarantee the precision of approximated values [6]. Interval iteration methods are proposed to guarantee the precision of computed values for the extremal reachability probabilities [4, 6] and for extremal expected rewards [2]. Two vectors $\underline{x}$ and $\underline{y}$ are utilized to approximate the lower and upper bound of the extremal expected reward values. In this case, vectors x and y can finally converge from below and above to $\underline{\mathbb{E}}^{max}$.

Considering ϵ for the precision of computations, the interval iteration method iterates until the maximum difference of values between two vectors for all states drops below 2ϵ, that is in an iteration k, if $\max_{s \in S^1_{min}}(y^k_s - x^k_s) < 2\epsilon$ holds. After termination we have $|\frac{y^k_s + x^k_s}{2} - \mathbb{E}^{max}_s| < \epsilon$ for each state $s \in S^1_{min}$. Algorithm 1 computes the interval iteration for the maximal expected rewards. To avoid some redundant computations, a modified version (called separate interval iteration) is proposed in [12].

Several methods are available for pre-computation to calculate the starting values for $\underline{x}$ and $\underline{y}$. In the case of non-negative rewards, $\underline{0}$ is trivially a starting point of $\underline{x}$. For $\underline{y}$, several techniques are presented in [2] to calculate the starting point. A main drawback of these techniques to compute an initial value for the vector $\underline{y}$ is that in come cases, the proposed initial values are far away from the exact values. This drawback increases the running time of computations. To have better initial values for $\underline{y}$ several techniques have been proposed in [10, 12, 15]. The idea of these techniques is to consider the computed values for the vector $\underline{x}$ to approximate the initial value for $\underline{y}$. In the proposed approaches in [10, 12], the standard value iteration

Algorithm 1 Interval iteration for $\mathbb{E}_s^{max}$.

```
input: an MDP $M = (S, s_0, Act, P, R)$, a set $G$ of goal states, the set $S^1_{min}$, a threshold $\epsilon$
and two initial vectors $\underline{x}$ and $\underline{y}$ for the lower and upper bound of values
output: Approximation of $\mathbb{E}_s^{max}$ for all $s \in S$ with the precision of $\epsilon$
repeat
  for all $s \in S^1_{min}$ do
    $x_s = \max\limits_{\alpha \in Act(s)} (R(s,\alpha) + \sum_{s' \in S} P(s,\alpha,s') \cdot x_{s'})$ ;
    $y_s = \min\{y_s, \max\limits_{\alpha \in Act(s)} (R(s,\alpha) + \sum_{s' \in S} P(s,\alpha,s') \cdot y_{s'})\}$ ;
  end for
until $max_{s \in S^1_{min}} (y_s - x_s) \leq 2\epsilon$;
return $(\frac{y_s + x_s}{2})_{s \in S^1_{min}}$;
```

method is applied to update the values of $\underline{x}$. After terminating value iteration, the initial value for $\underline{y}$ is computed by considering the computed values for $\underline{x}$. In this step and for each state $s \in S$, the proposed approach considers $y_s = c \cdot x_s$ where $c > 1$ is a constant. To check the soundness of the computed vector $\underline{y}$ several number of iterations of the VI method is applied on this vector. If after these iterations, the value of all states decrease, it is guaranteed that the computed vector is correctly an upper-bound of the values and can be used for the remainder computations. Otherwise, several other iterations are applied on the vector $\underline{x}$ and a higher value for c is considered. The process continues until getting a sound vector $\underline{y}$. More details about this approach is available in [10].

3. The Proposed Method

The possibility of using PI to update the values of the vectors $\underline{x}$ and $\underline{y}$ for the interval iteration method has been recently investigated in [9]. It is not trivial to apply the PI method to update the values of the upper-bound vector and [9] proposes an example where the PI method picks some non-optimal policies and the II method with it terminates with some values far away from the exact one. This case may happen for both extremal reachability probabilities and expected rewards. Instead, a hybrid approach is mentioned in [9] to use a combination of VI, PI and action elimination. The experimental evaluation of [9], however, does not cover this case and only reports the results for the optimistic version of II[10] where the standard VI is used to update values. It also considers PI when it uses II to resolve each induced DTMC. In this section, we provide our approach to apply PI as a solver for the interval iteration method and discuss the possibility of applying some advanced techniques to reduce the overall running time.

3.1. Sound Interval Iteration with Policy Iteration and Action Elimination

In this section, we explain how to use PI and VI with action elimination for the interval iteration method in a sound way. We propose this approach in Algorithm 2. It gets a MDP model M, a constant $K > 1$, a threshold ϵ for terminating the computations and an initial value for the vector $\underline{x}$. The threshold ϵ can be considered as is in VI, PI and II[5]. For the initial value of $\underline{x}$ the values for all states can be set to 0. For better explanation, we divide the algorithm to three steps. The first step, contains a while loop and follows an approach similar to the optimistic VI to compute a sound vector for upper-bound. However, our algorithm applies PI to update the states of the vector $\underline{x}$. After terminating PI, the initial value for the vector $\underline{x}$ is computed by considering the constants $c > 1$ and $d > 0$. Then the algorithm updates the values of $\underline{y}$ by applying K iterations of VI. If the value of all states decrease, the current values for $\underline{y}$ provide a correct upper bound. Otherwise, several additional iterations of PI is applied by considering smaller values for the threshold ϵ. It is guaranteed that these process will finally terminate while has computed a sound vector $\underline{y}$ for upper-bounds [10, 12].

The second step, for each state $s \in S$ considers the computed values for x_s and label those actions that give under-approximation for y_s. Such actions can not give a correct value for the upper-bound and should be disregarded for the remainder of the computations.

The third step updates the value of the vectors $\underline{x}$ and $\underline{y}$ until reaching a point where the difference of the upper-bound and lower-bound for all states is less than ϵ. It uses PI to update the values of the lower-bound and VI with non-disabled actions for the upper-bound.

Using PI to update the values of $\underline{x}$ and mark some actions as disabled can bring several opportunity to improve the performance of the II method. There are some cases where the standard version of PI outperform VI and hence, the former method can be a better option than the later. In addition, the modified and improved versions of PI as are discussed in [9, 12, 13] are faster than VI in most cases. Eliminating non-optimal actions avoids useless computations and reduces the overall running time of the computations. The soundness of this algorithm relies on the fact that PI will always converge to the true values from below if it uses sufficient

Algorithm 2 II with Policy Iteration for $\mathbb{E}_s^{max}$.

```
input: an MDP $M = (S, s_0, Act, P, R)$, a set $G$ of goal states, the set $S^1_{min}$, a threshold $\epsilon$
and an initial vectors $\underline{x}$ for the lower bound of values
output: Approximation of $\mathbb{E}_s^{max}$ for all $s \in S$ with the precision of $\epsilon$
flag = True;
while flag == True do
  Apply PI on the vector $\underline{x}$ considering $\epsilon$ for termination of computations.
  for all $s \in S$ do
    $y_s = c \cdot x_s + d$;
  end for
  iters = 1;
  while iters < K do
    for all $s \in S$ do
      $y_s = \min\{y_s, \max\limits_{\alpha \in Act(s)}(R(s,\alpha) + \sum_{s' \in S} P(s,\alpha,s') \cdot y_{s'})\}$ ;
    end for
    iters = iters + 1;
  end while
  if $\exists\ s \in S^1_{min}$ where $y_s$ has never decreased then
    $\epsilon = \epsilon/2$;
  else
    flag = false;
  end if
end while
for all $s \in S^1_{min}, \alpha \in Act(s)$ do
  if $x_s > R(s,\alpha) + \sum_{s' \in S} P(s,\alpha,s') \cdot y_{s'}$ then
    Mark $\alpha$ as disabled.
  end if
end for
repeat
  for all $s \in S^1_{min}$ do
    Update $x_s$ using PI;
    Update $y_s$ considering non-disabled Actions;
  end for
until $max_{s \in S^1_{min}}(y_s - x_s) \leq 2\epsilon$;
return $(\frac{y_s + x_s}{2})_{s \in S^1_{min}}$;
```

number of iterations, i.e., it never gets stuck in wrong state values. For the upper-bound the soundness is provided when we apply the standard VI method [15] or disregard those actions that do not surely give the optimal values.

4. Experimental Results

To analyse the running time of the proposed method in this paper with the standard one, we consider a set of standard case studies that are widely used in the previous works [5, 9]. We use the PRISM model checker [11] to run the experiments. We also implemented Algorithm 2 in PRISM. To do so, we consider the interval iteration [2] method (called II), sound value iteration [15] (called SVI) and optimistic value iteration wthod[10] (called OVI) from the previous works. We also consider Algorithm 2 with the standard PI method (PII), its modified version [14] (MPII) and its improved version (IPII)[13]. The results of the experiments are reported in Table1. For each method, we report the running time of iterative computations and the total number of computations. All reported times are in seconds.

Table 1

The running time of the evalutaed methods.

Model Name	Parameter	Number of	II		SVI		OVI		PII		MPII		IPII	
(parameters)	Values	states	time	iters	time	iters	time	iters	time	iterations				
Coin	4,6	63616	89.7	67791	48.6	37750	41.3	30123	42	42056	33	30158	32.8	30147
(K,N)	4,10	104576	475	191473	226	93589	184	78495	154	85345	138	78538	139	78545
	6,2	1258240	832	20811	509	13840	475	11624	663	31445	339	11642	342	11675
Zeroconf	6	798471	23.2	1491	12.6	747	10.7	635	11.7	647	11.5	1086	9.78	794
(K)	10	3001911	96	1546	49.6	850	40.9	695	38.8	695	39,7	695	40.2	697
	14	4427159	174	1822	76.6	812	69.7	746	63.1	746	67.2	782	749	64
Wlan	200	171542	15.4	6502	12.2	4709	11	4391	11.3	6092	9.24	4394	9.75	4719
(ttm)	800	409142	102	16509	96.2	15117	93.7	14668	85.4	17464	77.3	14671	78.6	14742
	1600	725942	426	35841	367	31482	343	29016	293	32003	286	29019	288	29022

In most cases, our proposed method with modified policy iteration works better than the optimistic value iteration method. In some cases, even the proposed method with the standard policy iteration is faster than OVI. Altough in these cases, the overall number of iterations increases, but because they consider only one action in most iterations, they are faster than OVI. There are also some cases where the proposed method with improved policy iteration outperforms the other cases.

5. Conclusion

In this work, we propose an approach to use PI for updating the values of states for the II method. Experimental results demonstrate that our proposed technique with modified policy iteration outperforms the best known previous work for optimistic value iteration. For the future works, we plan to consider several improvements for policy iteration to reduce the overall running time of the proposed technique.

References

1. Baier, Christel, and Joost-Pieter Katoen. Principles of model checking. MIT press, 2008.
2. Baier, Christel, Joachim Klein, Linda Leuschner, David Parker, and Sascha Wunderlich. "Ensuring the reliability of your model checker: Interval iteration for Markov decision processes." In International Conference on Computer Aided Verification, pp. 160-180. Cham: Springer International Publishing, 2017.
3. Baier, C., Hermanns, H., Katoen, JP. (2019). The 10,000 Facets of MDP Model Checking. In: Steffen, B., Woeginger, G. (eds) Computing and Software Science. Lecture Notes in Computer Science, vol 10000. Springer, Cham. DOI: 10.1007/978-3-319-91908-9_21
4. Br?zdil, Tom??, Krishnendu Chatterjee, Martin Chmelik, Vojt?ch Forejt, Jan K?et?nsk?, Marta Kwiatkowska, David Parker, and Mateusz Ujma. "Verification of Markov decision processes using learning algorithms." In Automated Technology for Verification and Analysis: 12th International Symposium, ATVA 2014, Sydney, NSW, Australia, November 3-7, 2014, Proceedings 12, pp. 98-114. Springer International Publishing, 2014.
5. Vojt?ch Forejt, Marta Kwiatkowska, Gethin Norman and David Parker. Automated Verification Techniques for Probabilistic Systems. In M. Bernardo and V. Issarny (editors), Formal Methods for Eternal Networked Software Systems (SFM'11), volume 6659 of LNCS, pages 53-113, Springer. June 2011.
6. Haddad, Serge, and Benjamin Monmege. "Interval iteration algorithm for MDPs and IMDPs." Theoretical Computer Science 735 (2018): 111-131. DOI: 10.1016/j.tcs.2016.12.003
7. S. Haddad and B. Monmege. Reachability in MDPs: Refining convergence of value iteration. In 8th International Workshop on Reachability Problems (RP), volume 8762 of Lecture Notes in Computer Science, pages 125137. Springer, 2014.
8. Kwiatkowska, Marta, David Parker, and Hongyang Qu. "Incremental quantitative verification for Markov decision processes." In 2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN), pp. 359-370. IEEE, 2011.
9. Hartmanns, Arnd, Sebastian Junges, Tim Quatmann, and Maximilian Weininger. "A practitioners guide to MDP model checking algorithms." In International Conference on Tools and Algorithms for the Construction and Analysis of Systems, pp. 469-488. Cham: Springer Nature Switzerland, 2023.
10. Hartmanns, Arnd, and Benjamin Lucien Kaminski. "Optimistic value iteration." In International Conference on Computer Aided Verification, pp. 488-511. Cham: Springer International Publishing, 2020.
11. Kwiatkowska, M., Norman, G., and Parker, D. (2011a). Prism 4.0: Verification of probabilistic real-

time systems. In International conference on computer aided verification, pages 585591. Springer

12. Mohagheghi, Mohammadsadegh, and Khayyam Salehi. "Accelerating Interval Iteration for Expected Rewards in Markov Decision Processes." In ICSOFT, pp. 39-50. 2020.
13. Mohagheghi, Mohammadsadegh, Jaber Karimpour, and Ayaz Isazadeh. "Improving modified policy iteration for probabilistic model checking." Computer Science 23, no. 1) (2022): 63-80.
14. Puterman, Martin L., and Moon Chirl Shin. "Modified policy iteration algorithms for discounted Markov decision problems." Management Science 24, no. 11 (1978): 1127-1137.
15. Quatmann, Tim, and Joost-Pieter Katoen. "Sound value iteration." In International Conference on Computer Aided Verification, pp. 643-661. Cham: Springer International Publishing, 2018.