Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Optimal schedulers vs optimal bases: An approach for efficient exact solving of Markov decision processes

Sergio Giro [1]

Department of Computer Science, University of Oxford, Oxford, OX1 3QD, UK

ARTICLE INFO

Keywords:
Probabilistic systems
Linear programming
Exact arithmetic
Optimal basis
Optimal adversaries
Optimal schedulers
Optimal policies

ABSTRACT

Quantitative model checkers for Markov decision processes typically use finite-precision arithmetic. If all the coefficients in the process are rational numbers, then the model checking results are rational, and so they can be computed exactly. However, exact techniques are generally too expensive or limited in scalability. In this paper we propose a method for obtaining exact results starting from an approximated solution in finite-precision arithmetic. The input of the method is a description of a scheduler, which can be obtained by a model checker using finite precision. Given a scheduler, we show how to obtain a corresponding basis in a linear programming problem, in such a way that the basis is optimal whenever the scheduler attains the worst-case probability. This correspondence is already known for discounted MDPs, we show how to apply it in the undiscounted case provided that some preprocessing is done. Using the correspondence, the linear programming problem can be solved in exact arithmetic starting from the basis obtained. As a consequence, the method finds the worst-case probability even if the scheduler provided by the model checker was not optimal. In our experiments, the calculation of exact solutions from a candidate scheduler is significantly faster than the calculation using the simplex method under exact arithmetic starting from a default basis.

1. Introduction

Model checking of Markov Decision Processes (MDPs) has been proven to be a useful tool to verify and evaluate systems with both probabilistic and non-deterministic choices. Given a model of the system under consideration and a qualitative property concerning probabilities, such as "the system fails to deliver a message with probability at most 0.05", a model checker deduces whether the property holds or not for the model. As different resolutions of the non-deterministic choices lead to different probability values, verification techniques for MDPs rely on the concept of *schedulers* (also called policies, or adversaries), which are defined as functions choosing an option for each of the paths of an MDP. Model-checking algorithms for MDPs proceed by reducing the model-checking problem to that of finding the maximum (or minimum) probability to reach a set of states under all schedulers [2].

Different techniques for calculating these extremal probabilities exist: for an up-to-date tutorial, see [10]. Some of them (for instance, value iteration) are approximate in nature. If all the coefficients in the process are rational numbers, then the model checking results are rational, and so they can be computed exactly. If the probabilities are to be used for further purposes (for instance, in a theorem prover used to check other properties of the system) the approximate probabilities might not be adequate, or they might even correspond to a suboptimal scheduler.

E-mail address: sergio.giro.ar@gmail.com.

[1] This work was supported by DARPA and the Air Force Research Laboratory under contract FA8650-10-C-7077. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of AFRL or the U.S. Government.

0304-3975/$ – see front matter © 2013 Published by Elsevier B.V.
http://dx.doi.org/10.1016/j.tcs.2013.08.020

However, exact techniques are generally too expensive or limited in scalability. Linear programming (LP) can be used to obtain exact solutions, but in order to achieve reasonable efficiency it is often carried out using finite precision, and so the results are always approximations. (We performed some experiments showing how costly it is to compute exact probabilities using LP without our method.) In addition, the native operators in programming languages like Java have finite precision: the extension to exact arithmetic involves significant reworking of the existing code.

We propose a method for computing exact solutions. Given any approximative algorithm being able to provide a description of a scheduler, our method shows how to extend the algorithm in order to get exact solutions. The method exploits the well-known correspondence between model-checking problems and linear programming problems [2], which allows to compute worst-case probabilities by computing optimal solutions for LP problems. We do not know of any similar approach to get exact values. This might be due to the fact that, from a purely theoretical point of view, the problem is not very interesting, as exact methods exists and they are theoretically efficient: the problem is that, in practice, exact arithmetic introduces significant overhead.

The simplex algorithm [3] for linear programming works by iterating over different *bases*, which are submatrices of the matrix associated to the LP problem. Each basis defines a *solution*, that is, a valuation on the variables of the problem. The simplex method stops when the basis yields a solution with certain properties, more precisely, a so-called *feasible* and *dual feasible* solution. By algebraic properties, such a solution is guaranteed to be optimal.

The core of our method is the interpretation of the scheduler as a basis for the linear programming problem. Given a scheduler complying with certain natural conditions, a basis corresponding to the scheduler can be used as a starting point for the simplex algorithm. We show that, if the scheduler is optimal, then the solution associated to the corresponding basis is feasible and dual feasible, and so a simplex solver provided with this basis needs only to check dual feasibility and compute the solution corresponding to the basis. As our experiments show, these computations can be done in exact arithmetic without a huge impact in the overall model-checking time. In fact, using the dual variant of the simplex method, the time to obtain the exact solution is less than the time spent by value iteration. If the scheduler is not optimal, the solver starts the iterations from the basis. This is useful for two reasons: we can let the simplex solver finish in order to get the exact solution; or, once we know that we are not getting the optimal solution, we can perform some tuning in the model checker as, for instance, reduce the convergence threshold (we also show a case in which the optimal scheduler cannot be found with thresholds within the 64-bit IEEE 754 floating point precision).

The correspondence between schedulers and bases is already known for discounted MDPs (see, for instance [8]). We show the correspondence for the undiscounted case in case some states of the system are eliminated in preprocessing steps. The preprocessing steps we consider are usual in model checking [10]: given a set of target states, one of the preprocessing algorithms removes the states that cannot reach the target, while the other one removes the states that can avoid reaching the target. These are qualitative algorithms based on graphs that do not perform any arithmetical operations.

The next section introduces the preliminary concepts we need throughout the paper. Section 3 presents our method and the proof of correctness. The experiments are shown in Section 4. The last section discusses related results concerning complexity and policy iteration.

2. Preliminaries

We introduce the definitions and known-results used throughout the paper, concerning both Markov decision processes and linear programming.

2.1. Markov decision processes

Definition 1. Let $\mathrm{Dist}(A)$ denote the set of discrete probability distributions over the set A. A Markov Decision Process (MDP) M is a pair (S, T) where S is a finite set of *states* and $\mathsf{T} \subseteq \mathsf{S} \times \mathrm{Dist}(\mathsf{S})$ is a set of *transitions*.[2] Given $\mu = (s, d) \in \mathsf{T}$, the value $d(t)$ is the probability of making a transition to t from s using μ. We write $\mu(t)$ instead of $d(t)$, and write $\mathrm{state}(\mu)$ for s. We define the set $\mathrm{en}(s)$ as the set of all transitions μ with $\mathrm{state}(\mu) = s$. For simplicity, we make the usual assumption that every state has at least one enabled transition: $\mathrm{en}(s) \neq \emptyset$ for all $s \in \mathsf{S}$.

We write $s \xrightarrow{\mu} t$ to denote $\mu \in \mathrm{en}(s) \wedge \mu(t) > 0$. A path in an MDP is a (possibly infinite) sequence $\rho = s^0.\mu^1.s^1.\cdots.\mu^n.s^n$ such that $s^{i-1} \xrightarrow{\mu^i} s^i$ for all i. If ρ is finite, the last state of ρ is denoted by $\mathrm{last}(\rho)$, and the length is denoted by $\mathrm{len}(\rho)$ (a path having a single state has length 0). Given a set of states U, we define $\mathrm{reach}(U)$ to be the set of all infinite paths $\rho = s^0.\mu^1.s^1.\cdots$ such that $s^i \in U$ for some i.

The semantics of MDPs is given by schedulers. A scheduler η for an MDP M is a function $\eta : \mathsf{S} \to \mathsf{T}$ such that $\eta(s) \in \mathrm{en}(s)$ for all s. In words, the scheduler chooses an enabled transition based on the current state. For all schedulers η, $t \in \mathsf{S}$, the set $\mathrm{Paths}(t, \eta)$ contains all the paths $s^0.\mu^1.s^1.\cdots.\mu^n.s^n$ such that $s^0 = t$, $\mu^i = \eta(s^{i-1})$ and $s^{i-1} \xrightarrow{\mu^i} s^i$ for all i. The reader familiar with MDPs might note that we are restricting to Markovian non-randomized schedulers (that is, they map states

[2] Defining transitions as pairs helps to deal with the case in which the same distribution is enabled in several states.

Please cite this article in press as: S. Giro, Optimal schedulers vs optimal bases: An approach for efficient exact solving of Markov decision processes, Theoretical Computer Science (2013), http://dx.doi.org/10.1016/j.tcs.2013.08.020