The BisimDist Library: Efficient Computation of Bisimilarity Distances for Markovian Models*

Giorgio Bacci, Giovanni Bacci, Kim G. Larsen, and Radu Mardare

Department of Computer Science, Aalborg University, Denmark
{grbacci,giovbacci,kgl,mardare}@cs.aau.dk

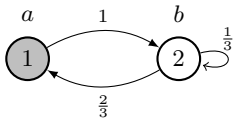
Abstract. This paper presents a library for exactly computing the bisimilarity Kantorovich-based pseudometrics between Markov chains and between Markov decision processes. These are distances that measure the behavioral discrepancies between non-bisimilar systems. They are computed by using an on-the-fly greedy strategy that prevents the exhaustive state space exploration and does not require a complete storage of the data structures. Tests performed on a consistent set of (pseudo)randomly generated instances show that our algorithm improves the efficiency of the previously proposed iterative algorithms, on average, with orders of magnitude. The tool is available as a Mathematica package library.

1 Introduction

Probabilistic bisimulation of Larsen and Skou [7] plays a central rôle in the verification of discrete-time Markov Chains (MCs), and this notion has been later extended to Markov Decision Processes with rewards (MDPs) [6]. Bisimulation equivalences may be used for comparing systems to a given model specification, or to make feasible the analysis of large systems by reducing their size by means of bisimilarity quotients. However, when the numerical values of probabilities are based on statistical samplings or subject to error estimates, any behavioral analysis based on a notion of equivalence is too fragile, as it only relates processes with identical behaviors. These problems motivated the study of *behavioral distances* (pseudometrics) for probabilistic systems, firstly developed for MCs [4,9,8] and later extended to MDPs [5]. The proposed pseudometrics are parametric in a *discount factor* $\lambda \in (0, 1]$ that controls the significance of the future in the measurement. These distances provide a way to measure the behavioral similarity between states and allow one to analyze models obtained as approximations of others, more accurate but less manageable, still ensuring that the obtained solution is close to the real one. These reasons motivate the development of algorithms for computing bisimilarity distances.

In [2] we proposed an efficient on-the-fly algorithm for computing the behavioral pseudometrics of Desharnais et al. [4] on MCs. Our method has been inspired by an alternative characterization of the pseudometric given in [3], that

* Work supported by the VKR Center of Excellence MT-LAB and the Sino-Danish Basic Research Center IDEA4CPS.




```
tm = MCtm[{{1,2}->1, {2,2}->1/3, {2,1}->2/3},2];
mc = MC[tm, {"a","b"}]
```

Fig. 1. Encoding of a Markov Chain as a data term in BisimDist.

relates the pseudometric to the least solutions of a set of equation systems induced by a collection transportation schedules. The pseudometric is computed by successive refinements of over-approximations of the actual distance using a greedy strategy that always chooses a transportation schedule that better improves the current approximation. This strategy avoids the exhaustive exploration of the state space, and has the practical advantage that allows one to focus only on computing the distances between states that are of particular interest. Experimental results have shown that this technique performs, on average, orders of magnitude better then the corresponding iterative algorithms proposed in the literature, e.g., in [3]. The algorithm in [2] has been recently adapted in order to compute the bisimilarity pseudometric introduced by Ferns et al. in [5] for MDPs with rewards (see [1] for a detailed account on this extension).

In this paper, we present the BisimDist library, composed of two Mathematica packages which implement our on-the-fly algorithm for computing the bisimilarity distances for MCs and MDPs, respectively. BisimDist is available at `http://people.cs.aau.dk/~giovbacci/tools.html` together with simple tutorials presenting use case examples that show all the features of the library.

2 The BisimDist Library

The BisimDist library consists of two Mathematica packages: MCDist and MDPDist providing data structures and primitives for creating, manipulating, and computing bisimilarity distances for MCs and MDPs respectively. It also has methods to identify bisimilarity classes and to solve lumpability problems.

The MCDist ***Package:*** An MC with n states is represented as a term of the form `MC[<tm>, <lbl>]`, where `<tm>` is an $n \times n$ probability transition matrix (`<tm>`$[\![i,j]\!]$ denotes the probability of going from the state i to the state j) and `<lbl>` is a vector of strings of length n (`<lbl>`$[\![i]\!]$ is the label associated with the state i). Note that states are implicitly represented as indices $1 \leq i \leq n$.

The probability transition matrices can be defined explicitly as a matrix, or implicitly by listing only the transitions which have nonzero probability by means of the function `MCtm` (see Fig. 1). Given a list `trRules` of rules of the form $\{i,j\} \to p_{i,j}$, the function `MCtm[trRules, n]` returns an $n \times n$ matrix where each pair (i,j) is associated with the value $p_{i,j}$, otherwise 0. An MC `mc` is displayed by calling `PlotMC[mc]`. Given a sequence $\texttt{mc}_1, \ldots, \texttt{mc}_k$ of MCs, `JoinMC[`$\texttt{mc}_1$`,...,`$\texttt{mc}_k$`]` yields an MC representing their disjoint union. The indices representing the set of states are obtained shifting the indices of the states

of the arguments according to their order in the sequence (e.g. if $\texttt{mc}_1$ has n states, the index corresponding to the i-th state of $\texttt{mc}_2$ in $\texttt{JoinMC[mc}_1\texttt{,mc}_2\texttt{]}$ is $n+i$).

Given an MC `mc` with n states, a list `Qpairs` of pairs of indices $1 \leq i, j \leq n$, and a rational discount factor $\lambda \in (0, 1]$, `BDistMC[mc, λ, Qpairs]` returns the list of all λ-discounted bisimilarity distances calculated between the pairs of states in `Qpairs` as list of rules of the form $\{i, j\} \to d_{i,j}$. The alias `All` is used for indicating the list of all pairs of states. `BDistMC` has the following options:

`Verbose:` (default `False`) displays all intermediate approximations steps;
`ConsistencyCheck:` (default `True`) checks that the term `mc` is a proper MC;
`Estimates:` (default `None`) takes a list of rules of the form $\{i, j\} \to d_{i,j}$ and computes the least over-approximation of the bisimilarity distance assuming $d_{i,j}$ to be the actual distance between the states i and j.

The package MCDist provides also the functions `BisimClassesMC`, which calculates the bisimilarity classes of an MC, and `BisimQuotientMC` that, for a given an MC, yields its quotient w.r.t. probabilistic bisimilarity.

The MDPDist ***Package:*** An MDP with n states and m action labels is represented as a term of the form `MPD[<tm>, <rw>, <act>]`, where `<tm>` is an $n \times m \times n$ labelled probability transition matrix (`<tm>` $[\![i,a,j]\!]$ is the probability of going from the state i to the state j, known that the action a as been chosen), `<rw>` is a $n \times m$ real-valued matrix representing a reward function, and `<act>` is a string-valued list of length m specifying the names of the action labels. States and action labels are implicitly encoded as indices.

Probability transition matrices of size $n \times m \times n$ can be defined by giving the nonzero transition probabilities as a list `trRules` of rules of the form $\{i, a, j\} \to p_{i,a,j}$ and calling `MDPtm[trRules, n, m]`. Analoguosly, $n \times m$ reward matrices can be defined by calling `MDPrm[<rwRules>, n, m]`, where `<rwRules>` is a list of rules of the form $\{i, a\} \to r_{i,a}$.

The MDPDist package is provided with an interface similar to MCDist with analogous semantics: `PlotMDP`, `JoinMDP`, `BDistMDP`, `BisimClassesMDP`, and `BisimQuotientMDP`.

3 Results and Conclusions

BisimDist is a research tool still undergoing development. While not yet mature enough to handle industrial case studies, the on-the-fly algorithm for computing the bisimilarity distance performs, on average, better than the iterative method proposed in [3]. Table 1 reports the average execution times of the on-the-fly algorithm run with discount factor $\lambda = 1/2$ on a collection of randomly generated MCs. We executed the iterative method on the same input instances, interrupting it as soon as it exceeded the running time of our method. The on-the-fly approach leads to a significant improvement in the performances: it yields the exact solution before the iterative method can under-approximate it with an error of ≈ 0.1, which is a non-negligible error for a value in the interval $[0, 1]$. A more detailed analysis of the performances and scalability can be found in [2].

Table 1. Comparison between the on-the-fly and the iterative methods on MCs.

# States	On-the-Fly (exact)	Iterative (approximated)		Approximation
	Time (sec)	Time (sec)	# Iterations	Error
10	1.003	1.272	3.111	0.0946
12	4.642	5.522	4.042	0.0865
14	6.336	7.188	4.914	0.1189
20	34.379	38.205	7.538	0.1428

The BisimDist library provides primitives that aid the analysis on probabilistic systems by reasoning in terms of approximate behaviors. In [1], we further improved the efficiency of the implemented on-the-fly algorithm on MDPs, also in relation to the addition of primitives for handling algebraic operations over probabilistic systems, such as synchronous/asynchronous parallel composition. We plan to apply similar on-the-fly techniques for computing bisimilarity distances on continuous-time probabilistic systems and timed automata.

References

1. G. Bacci, G. Bacci, K. G. Larsen, and R. Mardare. Computing Behavioral Distances, Compositionally. In K. Chatterjee and J. Sgall, editors, *38th Symposium on Mathematical Foundations in Computer Science*, volume ?? of *Lecture Notes in Computer Science*, page ?? Springer Berlin Heidelberg, 2013. To appear.
2. G. Bacci, G. Bacci, K. G. Larsen, and R. Mardare. On-the-Fly Exact Computation of Bisimilarity Distances. In N. Piterman and S. A. Smolka, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 7795 of *Lecture Notes in Computer Science*, pages 1–15. Springer Berlin Heidelberg, 2013.
3. D. Chen, F. van Breugel, and J. Worrell. On the Complexity of Computing Probabilistic Bisimilarity. In L. Birkedal, editor, *FoSSaCS*, volume 7213 of *Lecture Notes in Computer Science*, pages 437–451. Springer, 2012.
4. J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. Metrics for labelled Markov processes. *Theoretical Compututer Science*, 318(3):323–354, 2004.
5. N. Ferns, P. Panangaden, and D. Precup. Metrics for finite Markov Decision Processes. In *Proceedings of the 20th conference on Uncertainty in Artificial Intelligence*, UAI, pages 162–169. AUAI Press, 2004.
6. R. Givan, T. Dean, and M. Greig. Equivalence notions and model minimization in Markov decision processes. *Artificial Intelligence*, 147(1-2):163–223, 2003.
7. K. G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, 1991.
8. F. van Breugel, B. Sharma, and J. Worrell. Approximating a Behavioural Pseudometric without Discount for Probabilistic Systems. *Logical Methods in Computer Science*, 4(2):1–23, 2008.
9. F. van Breugel and J. Worrell. Approximating and computing behavioural distances in probabilistic transition systems. *Theoretical Computer Science*, 360(1-3):373–385, 2006.