An approximation algorithm for labelled Markov processes: towards realistic approximation

Alexandre Bouchard-Côté, Norm Ferns, Prakash Panangaden, and Doina Precup
School of Computer Science
McGill University
Montréal, Québec
Canada.
Email: alexandre.bouchard@mcgill.ca
{nferns, prakash, dprecup}@cs.mcgill.ca

***Abstract*— Approximation techniques for labelled Markov processes on continuous state spaces were developed by Desharnais, Gupta, Jagadeesan and Panangaden about 5 years ago. However, it has not been clear whether this scheme could be used in practice since it involves inverting a stochastic kernel. We describe a Monte-Carlo-based implementation scheme for this approximation algorithm. This is, to the best of our knowledge, the first implementation of this approximation scheme. The implementation involves some novel ideas about how to estimate infs using sampling and also replacing the explicit description of subsets of the state space by tests for membership. It is hoped that this work will enable more applications of continuous probabilistic LMP theory to emerge.**

I. Introduction

Labelled Markov processes (LMPs) are a theoretical formalism that generalizes both process algebra as well as traditional Markov chains. LMPs are processes that combine nondeterminism with probabilistic transitions. LMPs provide a foundation for interacting with discrete probabilistic systems. The interaction is synchronized on labels, just like in process algebras. There have been significant theoretical advances recently with the development of a notion of bisimulation for continuous LMPs, a logical characterization of bisimulation [1], metrics [2]–[5] and approximation theory [6]. The approximation theory developed by Desharnais, Gupta, Jagadeesan and Panangaden [6] has very appealing theoretical properties: it converges in the metric and also in the domain of LMPs and it captures exactly the logical properties of the original system in the limit. However, until now it has been unclear how to implement it in practice.

The initial study of labelled Markov processes in continuous state spaces was motivated by the potential for important practical applications in performance analysis and verification. This hope was based on the initial approximation schemes of Desharnais et. al. cited above as well as further improvements presented in [7]. Unfortunately, the development of concrete applications was slowed by obstacles that arose in the implementation of continuous state space approximation algorithms. Indeed, these algorithms are grounded in measure-theoretic ideas, some of which offer no direct algorithmic content. The biggest obstacle was that one had to invert an arbitrary measurable function. One could imagine that it was reasonable to assume conditions on the transition probability function—perhaps continuity or even piecewise linearity—in order to make progress. Indeed, a few examples were worked out by hand in this way. However, a general scheme to realize these approximations was lacking until now.

This paper describes a working implementation of the continuous state space approximation algorithm of [6]. This algorithm creates a finite state approximation of a given continuous process. The implementation relies heavily on techniques from probability theory, especially Monte Carlo methods, and eliminates the need for inverting exactly the transition probability function. The idea that Monte Carlo techniques could be useful was suggested in another paper on approximation [8], [9] where it was proposed that averaging should be used as an approximation method. However no concrete ideas were proposed there. The technique of the present paper has been developed for the approximation scheme of [6] rather than that of [8].

The first part of this paper summarizes the main elements of (continuous-state) labelled Markov process theory: bisimulation and approximation techniques. The reader interested in a in-depth exposition of these topics should refer to [6] and [3]. The key ideas on which the implementation is based are discussed in section III. More concrete implementation information can be found in section IV. Finally, some tests and applications of the code are described in section IV.

II. Background

In this section we give the basic background about LMPs and approximation. We assume that the reader is familiar with measure theory and elementary real analysis, as described, e.g., by Folland [10]. A basic reference for Monte Carlo methods is by Fox [11].

The basic mathematical object that we want to study is a model for *interactive probabilistic systems*:

Definition 1. *A* labelled Markov process $\mathcal{S}$ *is a triple* $(S, \mathfrak{M}, \boldsymbol{\tau})$*, where* S *is a state space,* $\mathfrak{M}$ *is a* σ*-algebra on* S[1] *and* $\boldsymbol{\tau}$ *is a family* $\{\tau_a : S \times \mathfrak{M} \to [0,1], a \in A\}$ *of transition sub-probability kernels indexed by a finite set of actions (or labels)* A.

The "choice" of actions dictates which kernel will be used to perform the transition from the current state to the next one (we use sub-probabilities to express disabled actions: action a_0 disabled at state s_0 is equivalent to $\tau_{a_0}(s_0, \cdot) \equiv 0$). This choice of action is made outside of the system description, and is external to the system. It could be, for instance, an agent picking actions so that it can reach some desirable state—this is the typical situation of interest in AI applications where the agents are following a policy—, or two distinct processes synchronizing their labels. We will call the object hence described an *LMP*.

The first tool we will use in our study of the structure of LMPs is that of *bisimulation*. Bisimulation for discrete systems was proposed and studied by Larsen and Skou [12]: the present definition is an adaptation to the continuous case where measure-theoretic conditions need to be imposed. Given a binary relation R on a set X we say that a subset Y is R-*closed* if $\{x \in X : \exists y \in Y. yRx\} \subseteq Y$; i.e. Y contains all points related by R to a point in Y. If R happens to be an equivalence relation then an R-closed set is a union of equivalence classes.

Definition 2. *We say that a binary relation* $R \subseteq S^2$ *on a LMP is a* bisimulation *if, for any* $s_1, s_2 \in S$ *with* $s_1 R s_2$*, and for any* R*-closed set* $X \in \mathfrak{M}$*, we have* $\tau_a(s_1, X) = \tau_a(s_2, X)\ \forall a \in A$*. We say that two states* s_1, s_2 *are bisimilar if there exists a bisimulation relation* R_0 *such that* $s_1 R_0 s_2$.

In particular, bisimulation gives us a way to compare two processes by first taking their *direct sum*: suppose $(S, \mathfrak{M}, \{\tau_a : S \times \mathfrak{M} \to [0,1], a \in A\})$ and $(T, \mathfrak{N}, \{\rho_a : T \times \mathfrak{N} \to [0,1], a \in A\})$ are two LMPs with initial states $s \in S$, $t \in T$. We merge the two LMPs into a new LMP constructed such that the new set of states U is the disjoint union of S and T, the new σ-algebra $\mathfrak{O}$ is generated by $\mathfrak{M} \cup \mathfrak{N}$, and the new kernels $\{\sigma_a : U \times \mathfrak{O} \to [0,1], a \in A\}$ are as follows: for all $s \in S, t \in T$, $X \in \mathfrak{M}$ and $Y \in \mathfrak{N}$, $\sigma_a(s, X \cup Y) = \tau_a(s, X)$ and $\sigma_a(t, X \cup Y) = \rho_a(t, Y)$. In the further discussions, ee will use whichever way of thinking is more convenient, depending on the context.

In a finite state space LMP [12], this definition corresponds to our intuition: two states s, t are bisimilar iff for all transitions from s to s', there is a transition t to t' with the same probability with s' and t' bisimilar, and vice-versa.

For the continuous case, bisimulation, as described above, is a generalization of this concept. The one-way counterpart of bisimulation is the concept of *simulation*:

Definition 3. *A reflexive and transitive relation (a preorder)* $R \subseteq S^2$ *on a LMP is a* simulation *if whenever* $s_1 R s_2$*, with* $s_1, s_2 \in S$*, we have that for all* $a \in A$ *and every* R*-closed measurable set* X*,* $\tau_a(s_1, X) \leq \tau_a(s_2, X)$

The central theorem provided by continuous probabilistic labelled Markov process theory is a characterization of bisimulation in terms of a surprisingly simple logic [1]:

Theorem 1. *Let* $\mathscr{L}_0$ *be the logic specified by the following grammar:*

$$\mathscr{L}_0 := T \big| \phi_1 \wedge \phi_2 \big| \langle a \rangle_q \phi,$$

where $a \in A$*,* $q \in \mathbb{Q} \cap [0,1]$ *and* $s \models \langle a \rangle_q \phi$ *is deemed to be true iff* $\exists X \in \mathfrak{M}$ *such that* $s' \models \phi\ \forall s' \in X$ *and* $\tau_a(s, X) > q$.

Then, two states s_1, s_2 *are bisimilar iff they satisfy the same formulas of* $\mathscr{L}_0$

This characterization is the basis of a collection of very useful tools, enabling quantitative analysis of LMPs. Simulation is characterized by a slightly extended logic:

Theorem 2. s_1 *is simulated by* s_2 *iff for all formulas* $\phi \in \mathscr{L}_\vee$*,* $s_1 \models \phi$ *implies* $s_2 \models \phi$*, where:*

$$\mathscr{L}_\vee := \mathscr{L}_0 \big| \bigvee_{i=1}^{\infty} \phi_i.$$

We now present the approximation scheme that we implemented.

Definition 4. *Let* $\mathscr{S} := (S, \mathfrak{M}, \boldsymbol{\tau} = \{\tau_a : S \times \mathfrak{M} \to [0,1], a \in A\})$ *be a LMP with starting state* s_0*. For* $n \in \mathbb{N}, \epsilon > 0$*, we define its* rational approximation*,* $\tilde{\mathscr{S}}_{n,\epsilon}(P, 2^P, \boldsymbol{\rho} = \{\rho_a : P \times 2^P \to [0,1], a \in A\})$*, as follow:*

- P *is a finite set whose elements are identified by a* level $\in \{0, 1, \ldots, n\}$ *and a measurable subset of* S*. At level 0, there is only one state,* $(S, 0)$*. The states at the other levels are defined inductively: given* $\{(C_1, l), (C_2, l), \ldots, (C_m, l)\}$*, the list of the* m *states of level* l*, we first partition the unit closed interval into disjoint half-open intervals of length* ϵ/m*, say* $(B_j)_{j \in I} = (\{0\}, (0, \epsilon/m], (\epsilon/m, 2\epsilon/m], \ldots, (1-\epsilon/m, 1])$*. Then, the states* $\{(D_k, l+1)\}$ *of level* $l+1$ *are obtained by letting* D_k *be the different subsets of* S *in the partition generated by the sets* $\tau_a(\cdot, C_i)^{-1}(B_j)$*, for every* $a \in A$*,* $i \in \{1, \ldots, m\}$ *and* $j \in I$*.*
- *The transitions in* $\tilde{\mathscr{S}}$ *occur only from states of level* $i+1$ *to states of level* i*. Transition probabilities are given by:*

$$\rho_a((X,k),(B,l)) := \begin{cases} \inf_{t \in X} \tau_a(t, B) \text{ if } k = l+1 \\ 0 \text{ otherwise.} \end{cases}$$

- *The initial state* p_0 *is the unique state* (X, n) *such that* $s_0 \in X$*.*

Example 3 illustrates the result of the application of this definition to a simple process.

[1]We actually require that S be an analytic space $(S, \mathscr{T})$ and that $\mathfrak{M} = \sigma(\mathscr{T})$. This is needed to prove, among other things, the logical characterization [1] used in this paper. It will not, however, be used explicitly, so we will not mention this condition in the rest of this paper. Analytic spaces are very general and cover the vast majority of spaces encountered in practical—or even impractical—situations.

This particular approximation scheme has very interesting properties with respect to the logic $\mathscr{L}_0$, and hence with respect to bisimulation:

Theorem 3. *For every $\epsilon > 0, n \in \mathbb{N}$, $\mathscr{S}$ simulates $\tilde{\mathscr{S}}$. More precisely, every state (X, l) of $\tilde{\mathscr{S}}$ is simulated in $\mathscr{S}$ by every $s \in X$.*

Theorem 4. *If a state $s \in S$ satisfies a formula $\phi \in \mathscr{L}_0$, then there is some approximation $\tilde{\mathscr{S}}_{n,\epsilon}$ such that $(X_s, n) \models \phi$, where X_s is the unique subset of S in the n-th level that contains s.*

These theorems tell us that every "finite piece of information", as embodied by a formula of the logic, is eventually captured by some approximant and that the approximants never describe any behaviour that is not a possible behaviour of the main system being approximated.

It has been argued that bisimulation—or indeed any equivalence relation—is not a robust notion in the presence of probabilities: a small perturbation of the probabilities will dramatically affect the relation. Accordingly metrics were introduced [2] as suggested first by Jou and Smolka [13]. These metrics measure behavioural "closeness" and when the processes are at zero distance[2] they are bisimilar. These metrics have a built-in discount factor so that actions in the future are discounted in their effect on measuring the distance between processes. We write d^c for the metric with discount factor c. The main theorem relating the metrics and the approximation is [3]:

Theorem 5. *If $\mathscr{S}$ involves a finite number of labels, $\tilde{\mathscr{S}}_{n,c^n/n}$ converges to $\mathscr{S}$ in the metric d^c with $c < 1$.*

The condition $c < 1$ is important in the calculation. However, it has been pointed out to us that the restriction to finite action sets could be weakened to countable sets if we change slightly the definition of the metric. The proof is in [14].

We have not introduced domains in the present paper but elsewhere [6] it has been shown that LMPs can be organized into a domain; it turns out that equality in the domain is bisimulation and that the order is simulation (essentially). The approximants constructed above form a directed set in this domain and their supremum gives the LMP being approximated.

III. Monte Carlo Techniques for LMP Approximation

The first question one must face before doing computation in a continuous state space is the representation problem: how should the transition probability kernels be expressed, assuming an uncountable range of values? In the case in which we have a "canonical" probability measure μ on $(S, \mathfrak{M})$ (for instance, in many problems this would be the Lebesgue measure), the most common solution to this question is to use a family of sub-probability density functions.

[2]Technically they are pseudo-metrics, distinct processes can be at zero distance.

Definition 5. A family of sub-probability density functions $f_a : S^2 \to [0, \infty)$, $s \in S, a \in A$, *is simply a family of* $(\mathfrak{M} \otimes \mathfrak{M})$*-measurable functions such that*

$$\int_S f_a(s_0, \cdot) d\mu \leq 1 \ \forall s_0 \in S, a \in A.$$

Then, the kernels are given by:

$$\tau_a(s_0, M) := \int_M f_a(s_0, \cdot) d\mu \ \forall M \in \mathfrak{M}, a \in A, s_0 \in S.$$

It is not hard to show that τ_a is then a labelled probability kernel (the fact that $\tau_a(s, \cdot)$ is a measure follows using the monotone convergence theorem, the measurability of $\tau_a(\cdot, M)$, using Fubini's theorem). We will denote this construction by $d\tau_a(s, \cdot) = f_a(s, \cdot) d\mu$. Recall that this representation is possible iff $\tau_a \ll \mu$ (by the Lebesgue-Radon-Nikodym theorem).

Example 1. *We now show a toy example of this construction that will be used later to test our algorithms. Consider a pair of 2-dimensional aquaria, arranged side by side horizontally. The first aquarium has horizontal coordinates $[0, \frac{1}{2}]$ and the second, $(\frac{1}{2}, 1]$. We are interested in the evolution of the horizontal position of a stochastic fish that starts its life in the first aquarium and has a choice of 2 actions:*

- swim *will change the position of the fish in its aquarium. The new position is drawn uniformly from the interval $[0, \frac{1}{2}]$.*
- jump *corresponds to an attempt to jump into the second aquarium. If the fish is at distance d from aquarium 2, it will fail and fall in the original aquarium with probability $2d$ (the next position will then be drawn uniformly from the interval $[\frac{1}{2} - d, \frac{1}{2}]$). If the fish succeeds, its new position is drawn uniformly from the interval $(\frac{1}{2}, 1]$. Unfortunately, the fish does not know that the second aquarium is filled with a liquid fatal for its metabolism. The death of the fish is modeled by disabling both actions in the second aquarium.*

Schematically:

$$\begin{tikzcd}
{[0, \frac{1}{2}]} \arrow[loop above, "{b[*]}"] \arrow[loop below, "{a[1]}"] \arrow[r, "{b[*]}"] & {(\frac{1}{2}, 1]}
\end{tikzcd}$$

where the label $a[p]$ on an edge (s_i, s_j) denotes that the probability to transition from s_i to s_j is p, given that action a is selected.

Note that the probability distribution induced by selecting action b is different for each state in $[0, \frac{1}{2}]$ from which the action is taken. This is denoted by $b[*]$. Hence, this LMP cannot be lumped into a finite state system.

Let $\mu =$ the Lebesgue measure on $[0, 1]$. We obtain easily that the kernels corresponding to the actions described above can be expressed using the following probability density func-

tions:

$$f_{\mathit{swim}}(x,y) := \begin{cases} 1 \textit{ if } x \in [0,\frac{1}{2}] \textit{ and } y \in [0,\frac{1}{2}] \\ 0 \textit{ otherwise} \end{cases}$$

$$f_{\mathit{jump}}(x,y) := \begin{cases} 2 \textit{ if } x \in [0,\frac{1}{2}] \textit{ and } y \in [x,\frac{1}{2}] \\ 4x \textit{ if } x \in [0,\frac{1}{2}] \textit{ and } y \in (\frac{1}{2},1] \\ 0 \textit{ otherwise} \end{cases}$$

Example 2. *Let us consider now a more realistic situation. Consider the onboard flight control system of a Cosmos-3MU launcher, a 2-stage, UDMH-fueled dispensable rocket often used to send small payloads into Earth orbit [15].*

A hypothetical problem for which the approximation scheme would be useful is the verification and/or evaluation of the effectiveness of flight guidance software for the Cosmos-3MU (In November 2000, and twice in January 2005, the second stage of the launcher failed to form the final orbit because of undiagnosed problems in this system. At least two commissions tried unsuccessfully to isolate the source of this "bug"). Suppose that the main controller must keep the launcher within distance r_{max} of the ideal trajectory by applying lateral speed corrections. The controller is composed of a sampling loop, the cyclic executive, that applies a thrust towards the ideal trajectory if needed. This loop structure motivates the discrete-time model of the problem. The state space is the cartesian product of the velocity space with the distance-to-trajectory space, $\mathbb{R}^2$ (with $r = -r, v = -v$). The actions are:

- actuate, *which applies a velocity correction towards the ideal trajectory. Due to the limited precision of these corrections, the result of this action from state (r_0, v_0) is modeled by a bivariate normal distribution centered at $\big(x_0 + \delta(v_0 - a_{\mathit{impulse}}), v_0 - a_{\mathit{impulse}}\big)$ with a strong, positive correlation such that the major axis of the elliptic isopleths of the density (that is, the locus of the points in the plane where $f_{\mathit{actuate}}(x,v) = c$) has a slope of $1/\delta$. The variance parameters can be set using the large amount of flight data available for this type of rocket (more than 400).*
- stay, *corresponds to the absence of velocity correction. It is modeled by a normal distribution f_{stay}, similar to f_{actuate} except that the center is at $\big(x_0 + \delta \cdot v_0, v_0\big)$.*

If, at any point in the stage-2 propulsion sequence, the controller fails to maintain the trajectory within distance r_{max} of the trajectory, a backup system takes control of the guidance. This is represented by disabling all actions. If, on the other hand, the controller successfully keeps the trajectory during 27 minutes, the orbit is reached and a special action, success, *is enabled to a special* success *state, s_{success}.*

Given that the set M, over which the density functions are to be integrated, can be an arbitrary measurable set, the next difficulty is to compute algorithmically these integrals. Numerical integration cannot be applied here because M could be "too nasty" geometrically to allow a nice partitioning. The solution comes from probability theory:

Lemma 1. *Let $(\Omega, \mathscr{F}, P)$, $(S, \mathfrak{M}, \lambda)$ be probability spaces. Assume that we can sample the random variables $X_1, X_2, \ldots$, $X_i : \Omega \to S$, identically and independently according to the distribution λ. Then, if $f : S \to \mathbb{R}$ is integrable and $M \in \mathfrak{M}$ we have:*

$$\frac{1}{n}\sum_{i=1}^{n}(\chi_M \cdot f) \circ X_i \to \int_M f d\lambda \ (a.s.).$$

This standard result is the basis of *Monte Carlo integration*. Its proof is fairly simple:

Proof: We have the following picture:

$$f \circ X_i : (\Omega, \mathscr{F}, P) \ \to \ (S, \mathfrak{M}, \lambda) \ \to \ (\mathbb{R}, \mathscr{B}_{\mathbb{R}})$$

Using the fact that $X_1, X_2, \ldots$ are independent and that f is measurable, we obtain easily, using Fubini's theorem, that $f \circ X_1, f \circ X_2, \ldots$ are independent. They are also clearly identically distributed and L^1, so we can apply Khinchine's Strong Law of Large Numbers to obtain:

$$\begin{aligned} \frac{1}{n}\sum_{i=1}^{n}(\chi_M \cdot f) \circ X_i \to \int_\Omega (\chi_M \cdot f) \circ X_1 dP \\ = \int_S \chi_M \cdot f dP_{X_1} \ (a.s.) \\ = \int_M f dP_{X_1} \\ = \int_M f d\lambda. \end{aligned}$$

■

The other operations on measurable sets and functions that we encountered in defintion 4 are:

1) Given a measurable set M and a measurable function f, compute the infimum of the value attained by f,
2) Given two measurable sets M_1, M_2, determine whether their intersection is non-empty (is-$\emptyset$: Sets $\to \{0,1\}$, is-$\emptyset(A) = 1$ iff $A = \emptyset$),
3) Given a measurable function f, compute the inverse image of an interval.

We will see in section IV how point 3 can be avoided. The basic idea is that, since we use Monte Carlo integration for the representation of the kernels, the only operation we need to impose on measurable sets is to test membership of a given point (in particular, there is no need for an operation that would express a measurable set as the union of intervals plus a null set, as it would be the case if we were using numerical integration, for instance). The two other points, however, have to be handled. Note that both arbitrary $\inf$'s and is-$\emptyset$'s cannot be computed algorithmically in general (the inf could be achieved on a set of measure zero), so we look for "measure-theoretic" equivalents that are computable (in a randomized computation model). For the case of $\inf$s, we use the following concept:

Definition 6. *Let $(X, \mathfrak{G}, \mu)$ be a measure space. We define the* essential infimum *over $M \in \mathfrak{G}$ of a bounded measurable function $f : (X, \mathfrak{G}) \to (\mathbb{R}, \mathscr{B}_{\mathbb{R}})$ to be:*

$$\mathit{ess}\inf_M f := \sup\Big\{a \in \mathbb{R} : \mu\Big(\big\{x \in M : f(x) < a\big\}\Big) = 0\Big\}.$$

Now suppose that $\mathscr{S} := (S, \mathfrak{M}, \boldsymbol{\tau})$ is a LMP such that $\tau_a \ll \mu$ for all kernel τ_a in $\boldsymbol{\tau}$ (in which case we will write "$\boldsymbol{\tau} \ll \mu$"), where μ is a measure on S from which we can sample points. In this situation, essential infima have the advantage of being computable:

Lemma 2. *Let $(\Omega, \mathscr{F}, P)$ be a probability space and assume that we can sample the random variables $X_1, X_2, \ldots,$ $X_i : \Omega \to M$, identically and independently according to the distribution μ, where $M \in \mathfrak{M}$ and $\mu(M) > 0$. Then if $f : S \to \mathbb{R}$ is bounded and measurable we have:*

$$\min\big\{f \circ X_i : 1 \leq i \leq n\big\} \to \textit{ess}\inf_M f \text{ (\textit{in P-probability})}.$$

Proof: First note that $\operatorname{ess\,inf}_M f < \infty$ if and only if $\mu(M) > 0$. Let $\epsilon > 0$ be given. By definition of supremum, we have that

$$p_0 := \mu\Big(\big\{s \in M : f(s) - \operatorname{ess}\inf_M f \leq \epsilon\big\}\Big)$$

must be positive. Then for any $i \in \mathbb{N}$,

$$\begin{aligned}
&P\Big(\big\{\omega \in \Omega : |f \circ X_i(\omega) - \operatorname{ess}\inf_M f| > \epsilon\big\}\Big)\\
&= \mu\Big(\big\{s \in M : |f(s) - \operatorname{ess}\inf_M f| > \epsilon\big\}\Big)\\
&= \mu\Big(\big\{s \in M : f(s) - \operatorname{ess}\inf_M f > \epsilon\big\}\Big)\\
&= 1 - p_0 < 1.
\end{aligned}$$

Hence, by the independence of the X_i's, the probability of the intersection of these events as $i \in \{1, 2, \ldots, n\}$ can be made arbitrarily small by sampling enough X_i's (i.e. by picking n large enough). ■

Similarly, we do not attempt to decide whether sets are empty, we rather restrict ourselves to deciding whether they have measure zero. This is done using the obvious Monte Carlo algorithm which returns true iff all the points sampled from the canonical measure μ do not belong to N. This clearly decides with high probability whether N has μ-measure zero or not. We shall call this algorithm is-null.

We now show that $\inf$'s and is-$\emptyset$'s can be replaced by $\operatorname{ess\,inf}$'s and is-null's in the rational approximation algorithm presented in section II, and that without altering the important properties possessed by the resulting approximations (theorems 3 and 4).

Theorem 6. *Let $\mathscr{S} = (S, \mathfrak{M}, \boldsymbol{\tau})$ be a LMP, $\boldsymbol{\tau} := \{\tau_a : S \times \mathfrak{M} \to [0,1], a \in A\}$, and $(S, \mathfrak{M}, \mu)$ be a probability measure from which we can sample points and such that $\boldsymbol{\tau} \ll \mu$. Assume also that the start state of $\mathscr{S}$ is μ-randomly selected (A property that is modeled in the following way: we add a state s_0 to S, which will be the starting state. The transition from s_0 to S is μ for all $a \in A$, and the transitions from $s' \in S$ to s_0 are all set to 0).*

Then for all $\epsilon \in \mathbb{Q}, \epsilon > 0$, $n \in \mathbb{N}$, the Monte Carlo rational approximation *$\tilde{\mathscr{Q}}_{\epsilon,n}$ (i.e., the approximation corresponding to definition 4 with* $\inf$*'s replaced by* $\operatorname{ess\,inf}$*'s and is-$\emptyset$ replaced by is-null) is computable and has the following properties:*

1) *every state (X, l) of $\tilde{\mathscr{Q}}_{\epsilon,n}$ is simulated in $\mathscr{S}$ by every $s \in X$,*
2) *if a state $s \in S$ satisfies a formula $\phi \in \mathscr{L}_0$, then there is some approximation $\tilde{\mathscr{Q}}_{\epsilon_0,n_0}$ such that $(X_s, n) \models \phi$,*
3) *given $c \in (0,1)$, let $\tilde{\mathscr{Q}}_n$ be $\tilde{\mathscr{Q}}_{\epsilon,n}$ with $\epsilon = c^n/n$. Then $\tilde{\mathscr{Q}}_n$ converges to $\mathscr{S}$ with respect to the metric d^c.*

Proof: The computability statement is already established by lemma 1 and 2.

To show 1, 2 and 3, we will use the following construction:

$$\begin{tikzcd}
\mathscr{S} \arrow[r, no head] \arrow[d, dashed] \arrow[dr, squiggly] & \mathscr{Q} \arrow[d, dashed] \\
\tilde{\mathscr{S}}_{\epsilon,n} & \tilde{\mathscr{Q}}_{\epsilon,n}
\end{tikzcd}$$

where the dashed lines denote approximation by the non Monte Carlo method, curved lines, approximation by the Monte Carlo method, and plain lines, bisimulation.

We first construct, for a given $\epsilon \in \mathbb{Q}, \epsilon > 0$ and $n \in \mathbb{N}$, a new LMP $\mathscr{Q} = (S \cap Z^c, \sigma(S \cap Z^c), \boldsymbol{\tau}|_{S \cap Z^c \times \sigma(S \cap Z^c)})$ which differs only on a set Z of μ-measure zero. We define this set Z as follows:

$$Z := \bigcup_{\mathscr{P}_{\epsilon,n}} \bigcup \big\{x \in X : \tau_a(x, B) < \operatorname{ess}\inf_{t \in X} \tau_a(t, B)\big\},$$

where $\mathscr{P}_{\epsilon,n} = (P, 2^P, \rho)$ ranges over all (non Monte Carlo) rational tree approximations of $\mathscr{S}$, and the inner union, over all $(X, l+1), (B, l) \in P$. It is easy to see that Z is a countable union of μ-null sets, and hence is indeed itself μ-null. It is easily seen that Z covers all the sets in S that cause a disagreement between the Monte Carlo approximation and the (standard) approximation. More precisely, by the way $\mathscr{Q}$ is constructed, we have:

$$\begin{aligned}
\tilde{\mathscr{Q}}_{\epsilon,n} &= \text{Monte Carlo approximation of } \mathscr{S}\\
&= \text{(non Monte Carlo) approximation of } \mathscr{Q},
\end{aligned}$$

where "=" stands here for equality of the probability transition matrices.

The next step is to show that $\mathscr{S}$ and $\mathscr{Q}$ are bisimilar, or equivalently, that for each $\phi \in \mathscr{L}_0$ and $s \in S \cap Z^c$, $s \models_{\mathscr{S}} \phi$ iff its copy in $\mathscr{Q}$ also satisfies ϕ (in $\mathscr{Q}$). The proof is by induction on the structure of formulas. The cases $\phi = T$ and $\phi = \psi_1 \cap \psi_2$, $\psi_1, \psi_2 \in \mathscr{L}_0$ are trivial, so suppose $\phi = \langle a \rangle_q \psi$, with $q > 0$ and $\psi \in \mathscr{L}_0$. If $s \in S \cap Z^c$ satisfies ϕ in $\mathscr{Q}$, then by the fact the state space of $\mathscr{Q}$ is included in the state space of $\mathscr{S}$, we have that the copy of s in the state space of $\mathscr{S}$ also satisfies ϕ. Conversely, suppose $s \in S$ satisfies ϕ in $\mathscr{S}$. Let $[[\psi]]_{\mathscr{S}}$ denotes the set of states in $\mathscr{S}$ that satisfy ψ. In particular, $\tau_a(s, [[\psi]]_{\mathscr{S}}) > q$. We thence have:

$$\begin{aligned}
\tau_a(s, [[\psi]]_{\mathscr{Q}}) &= \tau_a(s, [[\psi]]_{\mathscr{S}} \cap Z^c)\\
&= \tau_a(s, [[\psi]]_{\mathscr{S}}) - \tau_a(s, [[\psi]]_{\mathscr{S}} \cap Z)\\
&= \tau_a(s, [[\psi]]_{\mathscr{S}}) > q,
\end{aligned}$$

using the fact that $\boldsymbol{\tau} \ll \mu$.

Now that all the edges of the diagram are established, the theorem follows directly from theorems 1, 2, 3, 4, and 5. For instance, to establish 1, note that $\forall \phi \in \mathscr{L}_{\bigvee}$:

$$\begin{aligned}(X,l) \models_{\tilde{\mathscr{Q}}_{\epsilon,n}} \phi &\Longrightarrow (s \models_{\mathscr{Q}} \phi \quad \forall s \in X)\\ &\Longrightarrow (s \models_{\mathscr{S}} \phi \quad \forall s \in X),\end{aligned}$$

where the first implication is backed by theorem 3, and the second, by theorem 1. Since this is true for all $\phi \in \mathscr{L}_{\bigvee}$, 1 follows using theorem 2. ■

IV. Experiments

The aproximation scheme described in the previous section was implemented in the Java™ programming language. The structure of our object-oriented library mimics the measure-theoretic formulation of the theory. The most important types are the interfaces[3] *LMP*, *MeasurableFunction*, *MeasurableSet*, *Measure*, *State* and *TransitionKernel* and they compose in the standard way (e.g., given a *MeasurableSet*, a *Measure* can associate it with a positive real number, namely its measure). The most useful implementations of these interfaces are included in the package (*DensityKernel*, *DiscreteKernel*, *BorelMeasure*, etc, which behave as their names suggest).

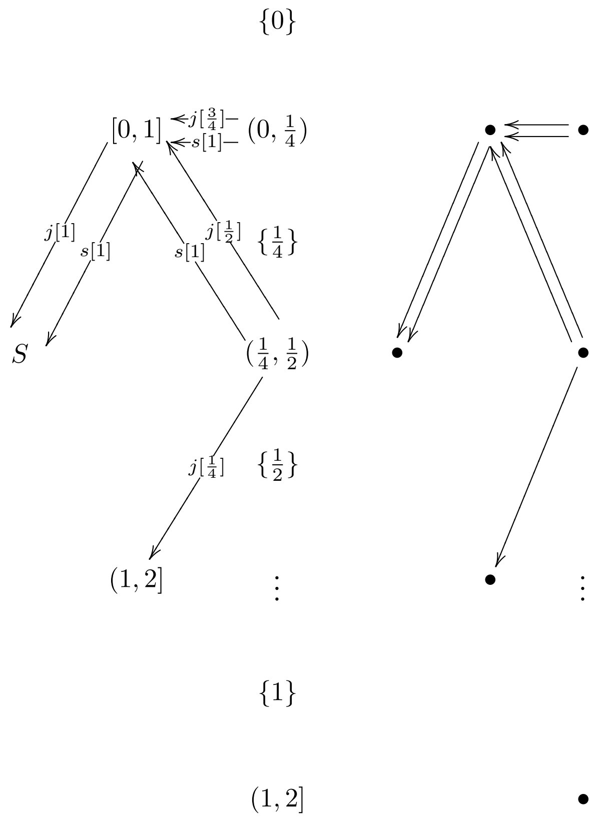
We summarize the structure of the class *LMPApproximator*, the core of our implementation of the Monte Carlo rational tree approximation. As mentioned earlier, it does not explicitly compute the inverse images of the form $\tau_a(\cdot, C_i)^{-1}(B_j)$. Instead, it uses an auxiliary class, *InverseSet*, which implements *MeasurableSet*. The only method specified by *MeasurableSet* is *contains(State s)*, which returns yes or no depending on whether or not a given measurable set contains state s. In this way *InverseSet* can be computed by simply maintaining C_i and B_j in memory, and given a state s, testing whether $\tau_a(s, C_i) \in B_j$. Note that the assumptions on *MeasurableSet* can be kept so simple (only one method is required) because we use Monte Carlo techniques rather than standard numerical methods (which often require creating a partition using more stringent assumptions on the geometry of the sets in consideration). Pseudocode for computing the approximant is shown in figure 1.

Note that *generatePartition*$(A_1, A_2, \ldots, A_n)$ can be implemented by computing, for each subset $\{i_1, i_2, \ldots, i_k\}$ of $\{1, 2, \ldots, n\}$, is-null$(A_{i_1} \cap A_{i_2} \cap \cdots \cap A_{i_k})$, and creating a block for each non-μ-null set. Observe, however, that this way of generating a partition is very inefficient. Fortunately, this can be easily optimized by using the inclusion properties of the sets in the approximation; indeed, at each successive level, the new partition forms a refinement of the previous one, so that many of the intersections in *generatePartition* can be declared μ-null without actually computing is-null. The fact that for each action, the generators already form a partition can also be used to speed-up considerably the asymptotic running time.

Similarly, additional important optimizations of the other Monte Carlo computations can be performed. This is particularly important when deep approximations (in n or ϵ) are needed, for in this case the set over which Monte Carlo integration or ess $\inf$ is carried out can very be small (in measure) compared to the whole space. This slows down convergence of the Monte Carlo methods because large numbers of points must be sampled before getting an "interesting" point, that is, one that belongs to the given set. However, once more through the inclusion of the successive partitions, it is possible to use a new measure λ in the Monte Carlo algorithms, $\mu \ll \lambda$, that is more dense around points that are sampled in the enclosing partition.

We note that for many applications, not all of the information in the tree is required. Therefore, another crucial optimization consists of computing certain properties of the approximation dynamically, that is, as they are requested by the user of the approximation. For instance, the transitions in the tree approximation do not need to be known to compute the state space of the tree. They can be computed on demand.

Example 3. *We come back to example 1 to illustrate how the Monte Carlo approximation algorithm behaves in practice. By the very simple form that the kernels have (continuous except for a finite set of points), it is easy, for this special case, to work out by hand the* $n = 3, \epsilon = 1/2$ *rational tree approximation. We compare it with the result of our algorithm:*



The left hand side represents the result of the manual computation, the right hand side, the one generated by the package. We see that their structure is the same, except for some sets of measure zero (which do not allow non-zero transitions). Moreover, a modest 200 iterations on the Monte Carlo operations yields a precision of 0.2 *on the transition probabilities.*

[3]We adopt the Java object-oriented terminology

```
APPROXIMATE($lmp, \epsilon, n$)
1   $approxStateSpace \leftarrow \emptyset$
2   $previousLevel \leftarrow \emptyset$
3   $approximationKernels \leftarrow$new$DiscreteKernel[lmp\,.\,labels\,.\,size]$
4   $startState \leftarrow$new$ApproximationState(lmp\,.\,stateSpace, 0)$
5   $previousLevel\,.\,add(startState)$
6   $approxStateSpace\,.\,add(startState)$
7   for $l \in \{1, \ldots, n\}$
8         do $generators \leftarrow \emptyset$
9            $m \leftarrow previousLevel\,.\,size$
10           $intervalPartition \leftarrow \{\{0\}, (0, \epsilon\,/\,m], (\epsilon\,/\,m, 2\,\epsilon\,/\,m], \ldots, (1 - \epsilon\,/\,m, 1]\}$
11           for $(C, l-1) \in previousLevel$ and $currentInterval \in intervalPartition$ and $a \in lmp\,.\,labels$
12                do for $currentInterval \in intervalPartition$ and $a \in lmp\,.\,labels$
13                       do $generators\,.\,add($
14                          $new\ InverseSet(lmp\,.\,transitionKernel(a).\,transitionFunction(C),$
15                          $currentInterval))$
16           $currentLevel \leftarrow generatePartition(generators)$
17           for $(X, l) \in currentLevel$ and $(B, l-1) \in previousLevel$ and $a \in lmp\,.\,labels$
18                do $approximationKernel[a].transition((X, l), (B, l-1)) \leftarrow$
19                   $\operatorname{ess\,inf}_X\ lmp\,.\,transitionKernel(a).\,transitionFunction(B)$
20           $approximationStateSpace \leftarrow approximationStateSpace \cup currentLevel$
21           $previousLevel \leftarrow currentLevel$
22  return $(approximationStateSpace, approximationKernels, startState)$
```

Fig. 1. Pseudocode for the algorithm.

Preliminary tests with a version of the package equipped with some of the optimizations described in the previous paragraph indicate that this precision can be greatly improved without too much difficulties.

Example 4. *How can this scheme be useful in concrete situations? Let us go back to example 2 now. An important quantity to assess in this case is the expected number of* actuate *actions required to complete the task. Because of the continuity of the state space, finite-state approximations are needed in order to carry out the integral. This illustrates the need for finite state approximation with good properties with respect to the metric introduced previously.*

Using our package and the densities described in the first part of this example, an approximation of small depth was easily and automatically constructed (in a matter of minutes). We are confident that simple optimizations will enable us to produce approximations that are deep enough to perform nontrivial validations and performance evaluations.

V. Related work

The problem of approximating a continuous system with a discrete one is of great interest in the control community as well, especially for people working on Markov Decision Processes (MDPs) and similar models. MDPs are very similar to LMPs, but they also allow a notion of reward for each label. The goal is usually to find an assignment of labels to states such that the total (discounted) reward obtained is maximized. In this case, fixed-resolution discretizations are very poor, since the long-term return may be very flat in part of the state space and very variable in others.

A variable-resolution heuristic discretization method has been proposed by [16]. They rely on a kd-tree data structure to maintain the approximate return values, and to compute new values quickly. They propose several discretization criteria and evaluate them empirically. The solution they obtain converges to the true value function as the size of the discretization cells approaches 0. In practice, they can handle continuous-state MDPs with states described by up to 10 dimensions. This result is considered state-of-the-art in the MDP literature. A discretization method based on a tree representation was also presented in [17]. They construct a tree that provides a discretization of the state space from trajectories of the system, generated using Monte Carlo sampling.

The similarity with our approach is that they also use Monte Carlo techniques to sample. In our case we use the sampling to estimate the inverse of the transition probability function while they are estimating the trajectories. In both cases we are looking at dynamical aspects of the systems rather than just the geometry of the state space as is sometimes done in robotics applications. However, we are trying to construct an approximate model whereas Munos and Moore are trying to estimate value functions and optimal policies. The goals and constructions are thus quite different.

VI. Conclusions and Future Work

This paper describes a concrete realization of the approximation scheme of Desharnais et. al. [6]. We emphasize that there were theoretical problems that needed to be solved; this paper does not merely describe a "program" to implement the algorithm of Desharnais et. al. The main problems were as follows.

- Stochastic kernels had to be inverted to compute the partitions of the state space. This was solved by not computing the inverse but rather implementing procedures for testing whether an element belonged to a particular block of the partition. This was done using sampling.
- To compute the inf of a transition probability function by sampling one has the danger that the inf is realized on a very improbable set and hence not seen in the sampling process. This was solved by using the essential inf and by the results that we proved showing that the approximations obtained this way also converge to the right result.
- The problem of deciding whether a given set is empty was solved similarly (using sampling, and testing μ-nullity instead).

The implementation was done in Java and preliminary experiments were carried out using this implementation. However, this implementation is a proof of concept and is not claimed to be ready for general use. We are very optimisitic that number of simple improvements will greatly improve the performance and is the subject of study in the coming weeks. We are also planning a large scale experimental investigation—involving a robotics motion planning application—with this improved version of the system. These results will be reported in a paper in preparation.

Acknowledgement

The authors have been supported by a grant from NSERC during the course of this work.

References

[1] J. Desharnais, A. Edalat, and P. Panangaden, "Bisimulation for labeled Markov processes," *Information and Computation*, vol. 179, no. 2, pp. 163–193, Dec 2002.

[2] J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden, "Metrics for labeled Markov systems," in *Proceedings of CONCUR99*, ser. Lecture Notes in Computer Science, no. 1664. Springer-Verlag, 1999.

[3] ——, "A metric for labelled Markov processes," *Theoretical Computer Science*, vol. 318, no. 3, pp. 323–354, June 2004.

[4] F. van Breugel and J. Worrell, "Towards quantitative verification of probabilistic systems," in *Proceedings of the Twenty-eighth International Colloquium on Automata, Languages and Programming*. Springer-Verlag, July 2001.

[5] ——, "An algorithm for quantitative verification of probabilistic systems," in *Proceedings of the Twelfth International Conference on Concurrency Theory - CONCUR'01*, ser. Lecture Notes In Computer Science, K. G. Larsen and M. Nielsen, Eds., no. 2154. Springer-Verlag, 2001, pp. 336–350.

[6] J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden, "Approximating labeled Markov processes," *Information and Computation*, vol. 184, no. 1, pp. 160–200, July 2003.

[7] V. Danos and J. Desharnais, "Labeled Markov Processes: Stronger and faster approximations," in *Proceedings of the 18th Symposium on Logic in Computer Science*. Ottawa: IEEE, 2003.

[8] V. Danos, J. Desharnais, and P. Panangaden, "Conditional expectation and the approximation of labelled markov processes," in *CONCUR 2003 - Concurrency Theory*, ser. Lecture Notes In Computer Science, R. Amadio and D. Lugiez, Eds., vol. 2761. Springer-Verlag, 2003, pp. 477–491.

[9] ——, "Labelled markov processes: Stronger and faster approximations," *Electronic Notes in Theoretical Computer Science*, vol. 87, pp. 157–203, November 2004.

[10] G. B. Folland, *Real analysis : modern techniques and their applications*. Wiley, 1999.

[11] B. L. Fox, *Strategies for quasi-Monte Carlo*. Kluwer Academic, 1999.

[12] K. G. Larsen and A. Skou, "Bisimulation through probablistic testing," *Information and Computation*, vol. 94, pp. 1–28, 1991.

[13] C.-C. Jou and S. A. Smolka, "Equivalences, congruences, and complete axiomatizations for probabilistic processes," in *CONCUR 90 First International Conference on Concurrency Theory*, ser. Lecture Notes In Computer Science, J. Baeten and J. Klop, Eds., no. 458. Springer-Verlag, 1990.

[14] J. Desharnais, "Labelled Markov processes," Ph.D. dissertation, McGill University, November 1999.

[15] "http://www.russianspaceweb.com/cosmos3.html," Web site.

[16] R. Munos and A. Moore, "Variable resolution discretization in optimal control," *Machine Learning*, vol. 49, pp. 291–323, 2002.

[17] W. T. B. Uther and M. M. Veloso, "Tree based discretization for continuous state space reinforcement learning," in *Proceedings of AAAI-98*, Madison, WI, July 1998.