On-the-Fly Algorithms for Bisimulation Metrics

***Abstract*—We study the problem of determining approximate equivalences in Markov Decision Proceses with rewards using bisimulation metrics. We provide an extension of the framework previously introduced in Ferns et al. (2004), which computes iteratively improving approximations to bisimulation metrics using exhaustive pairwise state comparisons. The similarity between states is determined using the Earth Mover's Distance, as extensively studied in optimization and machine learning. We address two computational limitations of the above framework: first, all pairs of states have to be compared at every iteration, and second, convergence is proven only under exact computations.**

We extend their work to incorporate "on-the-fly" methods, which allow computational effort to focus first on pairs of states where the impact is expected to be greater. We prove that a method similar to asynchronous dynamic programming converges to the correct value of the bisimulation metric. The second relaxation is based on applying heuristics to obtain approximate state comparisons, building on recent work on improved algorithms for computing Earth Mover's Distance. Finally, we show how this approach can be used to generate new algorithmic strategies, based on existing prioritized sweeping algorithms used for prediction and control in MDPs.

I. Introduction

Bisimulation has been used as the canonical equivalence in transition systems for analyzing behaviour, as well as for clustering equivalent states to produce a reduced version of a system. The probabilistic version of bisimulation was first analyzed in the context of labeled Markov chains [1], and later extended in the context of Markov Decision Processes with rewards [2], with the intention of helping to compute optimal policies in large systems.

Metrics, or more precisely, *pseudometrics*, based on bisimulation were invented over a decade ago [3], [4] with the hope that they would yield an approach for *approximate reasoning* about probabilistic systems. These methods were extended successfully to Markov Decision Processes (MDPs) [5] with rewards, which are a standard way of modelling multi-stage decision making in operations research and artificial intelligence [6], [7]. Bisimulation metrics for MDPs [5] allowed establishing bounds on the difference in the optimal long-term value of states, based on the metric distance between them. These results were the basis of several suggested algorithms for state aggregation (i.e., clustering similar states in order to reduce the size of the system) [5], policy transfer (i.e., taking the solution of a small system and extrapolating it for a large system) [8] and re-formulating either the state representation [9] or the action representation [10] to make large MDPs easier to solve. The key attractive feature of using bisimulation metrics in all these contexts is that they allow one to construct smaller/transformed problems that can be solved efficiently, while the solution obtained is still guaranteed to be close to the optimal one *at all states*. This is a very strong guarantee, which most other methods cannot provide [11].

Unfortunately, computing the metric is expensive, due to two main factors. First, at each iteration of the computation, *all* pairs of states must be compared; second, the metric for each pair of states must be computed exactly, based on the previous metric estimate. This leads to a computational complexity of $O(|S|^4 \log |S|)$ per iteration (using Orlin's algorithm [12] for minimum-cost flow to solve the required transportation problems, as will be discussed below), which is clearly prohibitive for large state spaces – precisely the type of problem in which using the metric to reduce the system is most helpful. In fact, this complexity exceeds that of usual dynamic programming algorithms for solving MDPs, such as value iteration and policy iteration, which are typically $O(|S|^3)$. So it is specious to motivate the use of bisimulation metrics on the grounds that they can be used to reduce the size of the state space for other analyses in which one might be interested. Nevertheless, if one can compute approximations to bisimulation metrics efficiently, it could still prove to be useful.

There are a couple of reasons for the high computational cost in the framework of [5]: (a) information about the magnitude of the metric is incorporated for each pair of states under a *strict* schedule: an additional update to one pair cannot be performed unless all other pairs have been updated in between. (b) The correctness of the framework is guaranteed only under *exact* computation of a costly transportation problem (which will be discussed later). Even if approximations to this problem have been provided – and heuristics can easily be designed – these do not fit the theoretical framework of [5].

Our main contribution is to provide a way to overcome these obstacles. It is not only an algorithmic question: the technique we use allows the intermediate steps *to not even compute a metric*. To do this, we re-worked the theoretical tools, including the all-important duality theorem.

- The first contribution of the paper is a strategy that allows one to focus on certain pairs of states at each iteration, where the changes are happening rapidly, so that the computational effort can be expended more effectively. This can be viewed as an *on-the-fly technique* [13].

On-the-fly techniques have been used with a lot of success in the systems and verification literature. Popular examples include garbage collection strategies for the SPIN verification system originating from work by Gerald Holtzman [14], [15], or bisimulation computation for finite deterministic transition systems [13]. A similar type of strategy has been used with great success in the artificial intelligence and optimization literature to solve large decision problems; here, such methods

are called asynchronous or distributed dynamic programming [16], [17], [18].

A result of doing "on-the-fly" computation is that the framework does not rely on the metric property of the partial results anymore; *in the limit*, however, we will compute the same bisimulation metric.

- This leads to the second contribution of the paper: approximate computation of the updates to the metric estimate, relaxing the second restriction described above. We also provide new theorems that guarantee that one still gets the bisimulation metric in the limit.

The computation of the pseudometric analogue of bisimulation relies on the Earth Mover's Distance, a type of transportation distance[1] to compare probability measures over the state space. Exact computation of the metric is important to maintain Kantorovich duality [19], a crucial property in keeping faithful to the framework of partition-refinement strategies: any metric over the state space determines a partition of that space and every iteration of the algorithm provides an increase in all values of the metric, giving a refinement of the previous partition. In our work, we relax this constraint using "on-the-fly" computation of the bisimulation metric. Improving approximations to an exact bisimulation metric are computed, but now heuristics which incorporate domain information, and which have been studied and shown to provide good approximations [19], [20], [21], can be incorporated in the process of metric computation.

In this paper, we develop the ideas, for concreteness, in the context of Markov Decision Processes with rewards, but the all the theoretical results and algorithmic suggestions apply equally well to labelled Markov processes [22], in which there are no rewards.

The paper is structured as follows. In Section II we review the framework of Markov Decision Processes and the dynamic programming approach to solving them. Sections III and IV review existing methods to compute bisimulation metrics for MDPs, as well as methods for computing Earth Mover's Distance metrics. Section V introduces new theoretical results on a generalized framework in which bisimulation metrics can be computed under a flexible schedule and with approximate transportation metric computations. Section VI presents different algorithmic strategies that build on the theoretical results. Section VII presents empirical results, illustrating the computational savings of our approach. Finally, Section VIII concludes and presents avenues for future work.

II. Markov Decision Processes

A. Problem definition

A (finite) discounted Markov Decision Process (MDP) with rewards[2] is represented as a tuple $\langle S, A, P, R, \gamma \rangle$. A system transitions between states in the set S, under the effect of actions in the set A. The transitions are governed by a probability distribution $P : (S \times A) \to (S \to [0,1])$; we write $P_{sa}^{s'} \in [0,1]$ (instead of $P(s,a)(s')$) to denote the conditional probability of a transition to state s' given current state s and action a. This notation allows one to write many relevant quantities in "matrix-like" form, which is convenient for algorithms working with finite-state MDPs. For each s and a, $\sum_{s'} P_{sa}^{s'} = 1$. The rewards are given by $R : (S \times A) \to [0,1]$; without loss of generality, we assume that rewards are non-negative and bounded by 1. We denote by R_{sa} the reward for state s and action a. Lastly, $\gamma \in (0,1)$ is a discount factor used to emphasize more rewards received right away, compared to those received in the distant future.

A *policy* $\pi : S \to (A \to [0,1])$ is a distribution over the actions given the current state; we denote by π_s^a the conditional probability of choosing action a given that the current state is s. The policies we consider are Markovian (memoryless) and probabilistic; note that since the systems themselves are Markovian, the state is a sufficient statistic for the future behaviour and strategies that use the past history do not help. For each s,$\sum_a \pi_s^a = 1$.

Let s be an starting state for the random system, and (s_t, a_t) the random state and action at time step t, generated by choosing actions according to a fixed policy π, and following the environment's state dynamics P. The *value* of state s under π is defined as:

$$V^\pi(s) = E[R_{sa_1} + \cdots + \gamma^{t-1} R_{s_t a_t} + \ldots | a_t \sim \pi_{s_t}, s_{t+1} \sim P_{s_t,a_t}]$$

Value functions are important because they allow one to compare different policies. In particular, in finite MDPs, there exists at least one *optimal policy* π^*, whose value is best at *all* states (i.e., $V^\pi(s) \leq V^{\pi^*}(s)$ for all s and for any other π). For simplicity, we will denote V^{π^*} by V^*. The goal of solving an MDP is to find such an optimal policy, and computing value functions is an important intermediate step in achieving this goal. We note that the optimal value function V^* is unique in a finite MDP.

B. Iterative methods for computing value functions

Based on the definition above, it is easy to show that the value function V^π obeys the following set of linear equations, also known as *Bellman Equations*:

$$V^\pi(s) = \sum_a \pi_s^a \left(R_{sa} + \gamma \sum_{s'} P_{sa}^{s'} V^\pi(s') \right), \forall s \in S \quad (1)$$

A popular approach to solving the system uses fixed point iterative methods similar to the *Jacobi method*:

1) for every $s \in S$, initialize $V_0(s) = 0$; $k = 1$
2) repeat:

$$V_k(s) = \sum_a \pi_s^a \left(R_{sa} + \gamma \sum_{s'} P_{sa}^{s'} V_{k-1}(s') \right) \quad (2)$$

until $\max_s |V_k(s) - V_{k-1}(s)| < \epsilon$, where ϵ is a desired precision parameter.

More details can be found in [6].

[1] Also known as the Kantorovich metric, the Wasserstein (or Vaserstein) metric, the Monge-Kantorovich metric and the Hutchinson metric.

[2] In artificial intelligence and operation research, all MDPs have rewards.

A similar approach can be used to solve the *Bellman optimality equations* which govern the optimal value function:

$$V^* = \max_a \left(R_{sa} + \gamma \sum_{s'} P_{sa}^{s'} V^*(s') \right), \forall s \in S \qquad (3)$$

Although (3) is a non-linear system, similar fixed point iterations can be designed and shown to converge to its solution. Let T denote the Bellman operator over value functions:

$$(TV)(s) = \max_a \left(R_{sa} + \gamma \sum_{s'} P_{sa}^{s'} V(s') \right) \qquad (4)$$

It is easy to show that T is a contraction, whose unique fixed point is V^*: $TV^* = V^*$. Hence, $V^* = \lim_{n\to\infty} T^n(0)$

C. Asynchronous Value Iteration

Traditional dynamic programming algorithms update all states at every iteration, which can be inefficient for large state spaces, especially if most rewards are 0. Asynchronous/distributed dynamic programming algorithms relax this requirement, allowing more flexible update schedules.

1) Gauss-Seidel methods: A popular improvement of the Jacobi methods in solving linear equations is known as the *Gauss-Seidel method*[23]. This exploits the well-known LU factorization to allow a modification to Equation (2): in updating $V_k(s)$, one can use $V_k(s')$ instead of $V_{k-1}(s')$, for all s' for which V_k was already computed. As shown in [24], this strategy can be extended to the non-linear system in (4). Moreover, Gauss-Seidel scheduling based on a fixed ordering of the states in S is not necessary: updates can be performed according to any schedule that guarantees that all states will be selected infinitely often.

2) Prioritized Sweeping: Many strategies can be used to determine schedules for updating states in asynchronous algorithms. For example, in reinforcement learning, states are typically sampled from trajectories generated according to randomized policies; [25] provides a comprehensive review of such methods. For our work, however, we focus on a heuristic called *prioritized sweeping*, which uses the magnitude of changes in the value function to determine the state update schedule [26]. The main idea is that if the value of a state has changed a lot, its "predecessors" (i.e., the states which transition to it) will also be subject to a big change in the next iteration. Hence, if such states are given higher priority for updating, the value function should converge quicker to good estimates.

We describe below the algorithm, which uses a priority queue Q of states for scheduling:

1) For all states $s \in S$, let $V_0(s) = 0$
2) For $k = 0, 1, 2, ...$
   a) If queue is empty, initialize Q with a state of choice
   b) Remove the top state s from the queue.
   c) $V_k(s) = (TV_{k-1})(s)$, and $\Delta = |V_k(s) - V_{k-1}(s)|$
   d) $V_k(s') = V_{k-1}(s')$ for all $s' \neq s$
   e) Add all states s' which can transition to s to the queue with priority:
   $$\Delta \sum\nolimits_a \pi_{s'}^a P_{s'a}^s$$
   If the states were already in the queue, their priority is updated to the maximum between the value above and their current priority.
   f) Stop when the maximum change over all states is below a desired threshold.

This approach has been used successfully both in large discrete MDPs, as well as in continuous robotics domains, e.g. [27], [28].

III. Bisimulation Metrics for MDPs

Suppose we are given an MDP and a partition $\mathcal{S}$ of the state space S; this partition determines a *probabilistic bisimulation* relation $\langle \cdot, \cdot \rangle_{\mathcal{S}}$, in which two states are related if they are in the same partition, under the following conditions:

$$\langle s, s' \rangle_{\mathcal{S}} \iff \forall a,\ (R_{sa} = R_{s'a} \text{ and } \forall S'' \in \mathcal{S},\ \sum_{s'' \in S''} P_{sa}^{s''} = \sum_{s'' \in S''} P_{s'a}^{s''})$$

One can generate an aggregated MDP based on $\mathcal{S}$, $M_{\mathcal{S}} = \langle \mathcal{S}, A, \bar{P}, \bar{R}, \gamma \rangle$ with

$$\forall S', S'' \in \mathcal{S}, \forall a \in A,\ \bar{R}_{S'a} = \frac{1}{|S'|} \sum_{s' \in S'} R_{s'a} \text{ and}$$

$$\bar{P}_{S'a}^{S''} = \frac{1}{|S'|} \sum_{\substack{s' \in S' \\ s'' \in S''}} P_{s'a}^{s''}$$

The relaxation from a bisimulation relation to a metric and subsequent bounds relating this metric to value functions have been derived in [5], based on a partition refinement algorithm: start with a metric that equates all states, $d_0 = 0$(i.e. start with a single cluster) and iteratively apply the following map F on the distance metric:

$$F(h)(s, s') := \max_a \left(|R_{sa} - R_{s'a}| + \gamma \mathcal{T}(h, P_{sa}, P_{s'a}) \right) \quad (5)$$

where $\mathcal{T}$ is known as the Monge-Kantorovich optimal transportation problem [19]. We will provide computational details of this update in the next section. An important theoretical result is that the above functional F has a fixed point d^*: $F(d^*) = d^*$, which is a bisimulation metric (i.e., a metric whose kernel is the strong bisimulation relation) and the procedure just described converges to this fixed point:

$$\lim_{n\to\infty} F^n(0) = d^*$$

In addition to the partition refinement algorithm, Ferns et al. [5] state bounds which allow one to assess the quality of aggregate MDPs obtained not only using bisimulation metrics, but also through other heuristics. Given any partition $\mathcal{S}$ (not necessarily a bisimulation relation), the optimal value function of the aggregate MDP $M_{\mathcal{S}}$ (in which states in a partition are forced to take the same action) and the optimal value

function of the original MDP are related through the following inequality [5]:

$$|V^* - V^*_{\mathcal{S}}| \leq \frac{\gamma}{(1-\gamma)^2} \max_{S' \in \mathcal{S}} \max_{s \in S'} \frac{1}{|S'|} \sum_{s' \in S'} d^*(s, s') \quad (6)$$

where d^* is the bisimulation metric. Note that the bound above is minimized by choosing partitions that put together states that are "close" in terms of the bisimulation distance d^*, i.e, states that are close to being bisimilar. Moreover, the bound is 0 if the aggregation respects the bisimulation relation faithfully.

IV. Earth Mover's Distance Metric

As mentioned above, computing the bisimulation metric relies on the Monge-Kantorovich, or Earth Mover's Distance (EMD) metric for comparing probability distributions. The **Monge-Kantorovich's optimal transportation problem** allows one to use a cost function $h : S \times S \to \mathbb{R}_+$ to obtain a numerical value characterizing the difference between probability maps P, Q. The optimization is performed over the following set :

$$\Pi(P, Q) = \left\{ \pi \geq 0 \mid \sum\nolimits_i \pi_{ij} = Q(j) \; ; \; \sum\nolimits_j \pi_{ij} = P(i) \right\}$$

and the objective is

$$\mathcal{T}(h, P, Q) := \inf_{\pi \in \Pi(P,Q)} \sum_{i,j} \pi_{ij} h(i, j) \quad (7)$$

Note that the definition applies with any cost function h, although the iterative fixed-point framework presented in Section III applies only to metrics (i.e., $F^n(0)$ is a metric for any $n \geq 0$). The main advantage of maintaining metric costs (and the crucial property in proving Equation (6)) is that it allows the *Kantorovich-Rubinstein duality* property, which means that one can compute another optimization problem instead. The **Kantorovich probability metric** is obtained from solving the following optimization:

$$\mathcal{K}(h, P, Q) := \sup_{f \in L(h)} \sum_i (P(i) - Q(i)) f_i$$
$$L(h) = \{ f : S \to [0, 1] \mid f_i - f_j \leq h(i, j) \; \forall i, j \}$$

Theorem 1. [Kantorovich-Rubinstein] *For any P, Q, if h is a bounded pseudo-metric, then*

$$\mathcal{T}(h, P, Q) = \mathcal{K}(h, P, Q)$$

A second important design quality of the fixed-point algorithm is that the intermediate metrics $F^n(0)$ can be used to generate intermediate partitions, which albeit not bisimulation relations, still provide specifications which are "close enough" to the original MDP. More details can be found in [5].

The transportation problem described in this section has been extensively studied, mostly in the domain of optimization, and also for popular special cases (e.g., quadratic cost over $\mathbb{R}^n$)[29], [19]. Moreover, the finite state space transportation problem becomes a linear program, so different exact methods have also been proposed [12], [30]. Recently, it has been raising interest in machine learning as well, where special cases have also been studied, especially for applications in computer vision [31], [32]. Here, we are not concerned with special cases, but focus on ideas for generating efficient modifications of general problems. For example, [21] proposes a threshold modification to a given metric to reduce the number of transportation ways between the source and the target, if this number is large. This strategy clearly provides lower bounds, but, as we will see in the next section, becomes useful because it does not rely on the metric property of the cost function.

V. A general framework for bisimulation metric computation

We proceed to generalize the approach of [5] to a more flexible procedure, which allows one to incorporate heuristics and different types of information about the state space both in scheduling the updates, as well as by allowing approximate update rules.

For notational simplicity, in this section we will work with symmetric maps $h : S \times S \to \mathbb{R}_+$(i.e. $\forall (s, s') \; h(s, s') = h(s', s)$ and $h(s, s) = 0$). We can emulate this by working with functions applied only on the following set: $\Omega_S := \{(s_i, s_j) : i < j\}$, where $\{s_i\}_{i=1}^{|S|}$ is any enumeration of S.

To accommodate a flexible schedule, we partition the state space at each time step k into three subsets: $S = \alpha_k \cup \beta_k \cup \delta_k$. The subset α_k contains the pairs that will be updated *without broadcasting* the new information: all recursive calls are based on the same function. The subset β_k contains all pairs that will *wait for broadcasting*: i.e. not be updated at time-step k. Finally, the subset δ_k contains the remaining pairs which will provide *noisy broadcasting* at the end of time-step k. This is done using any update function $\hat{h}_k$ (which might depend on h_{k-1}). We will show that if the latter is properly chosen, then one can obtain convergence to the same bisimulation metric as the original restrictive framework.

The pseudocode of the algorithmic approach we will follow is given below:

1) Initialize $h_0 = 0$
2) For $k = 1, 2, ...$
   a) Perform the update

$$h_k(s, s') := \begin{cases} F(h_{k-1})(s, s') & \text{if } (s, s') \in \alpha_k \\ h_{k-1}(s, s') & \text{if } (s, s') \in \beta_k \\ \hat{h}_k(s, s') & \text{if } (s, s') \in \delta_k \end{cases}$$

   until a desired convergence criterion is reached.

We now proceed to show that, under the proper choices in the approximation $\hat{h}$ and the scheduling strategy, the above framework will converge to d^* as described in Section III. For this, we will first prove a few important properties related to the functional F presented in Equation (5).

Lemma 1. *Consider two functions $h, h' : S \times S \to \mathbb{R}_+$ such that $h \leq h'$ (i.e., $h(s, s') \leq h'(s, s') \; \forall s, s'$). Then $\mathcal{T}(h, P, Q) \leq \mathcal{T}(h', P, Q)$, for any probability measures P, Q.*

Proof: Fix two probability measures P, Q, and let $\hat{\pi} \in \Pi(P, Q)$. Then,

$$\mathcal{T}(h, P, Q) \leq \sum_{i,j} \pi_{ij} h(s_i, s_j) \leq \sum_{i,j} \pi_{ij} h'(s_i, s_j), \text{since } \pi \geq 0$$

Note that the above holds for all $\hat{\pi} \in \Pi(P, Q)$, so we can conclude that

$$\mathcal{T}(h, P, Q) \leq \mathcal{T}(h', P, Q)$$

■

Lemma 1 describes a simple property of the EMD metric: if the costs of the transportation problem are increased, then the corresponding optimal transport strategy is more expensive. We will see in the next Lemma how this induces the monotonicity of F. The latter will prove to be a crucial property in proving that the algorithm is still convergent, even when the metric property is not enforced.

Lemma 2. *F is a monotone map: if*

$$\forall s, s', \quad h(s, s') \leq \bar{h}(s, s')$$

then:

$$\forall s, s', \quad F(h)(s, s') \leq F(\bar{h})(s, s')$$

Proof: Fix a tuple (s, s', a), then:

$$\begin{aligned}
&|R_{sa} - R_{s'a}| + \gamma \mathcal{T}(h, P_{sa}, P_{s'a}) \\
&\quad \leq |R_{sa} - R_{s'a}| + \gamma \mathcal{T}(\bar{h}, P_{sa}, P_{s'a}) \\
&\quad \leq \max_a (|R_{sa} - R_{s'a}| + \gamma \mathcal{T}(\bar{h}, P_{sa}, P_{s'a})) \\
&\Rightarrow \max_a (|R_{sa} - R_{s'a}| + \gamma \mathcal{T}(h, P_{sa}, P_{s'a})) \\
&\quad \leq \max_a (|R_{sa} - R_{s'a}| + \gamma \mathcal{T}(\bar{h}, P_{sa}, P_{s'a}))
\end{aligned}$$

■

The monotonicity of F has one obvious result: the framework just described will provide a monotonic increase towards the exact bisimulation metric. Importantly, we also want to make sure that "over-shooting" will not occur. We present below the first restriction on the approximate updates, aimed to ensure this property: $\hat{h}_k \leq d^*$.

Lemma 3. *Let $k > 0$. Assume that $\hat{h}_{k'} \leq d^*$ for all $k' \leq k$. Then $h_k \leq d^*$.*

Proof: We will use induction over k. Since h_0 is 0, the statement clearly holds. Now, assume $h_k \leq d^*$. Then, for $(s, s') \in \alpha_{k+1}$,

$$\begin{aligned}
h_{k+1}(s, s') &= F(h_k)(s, s') && \text{by definition} \\
&\leq F(d^*)(s, s') && \text{by monotonicity of } F \\
& && \quad \text{on the induction hypothesis} \\
&= d^*(s, s') && \text{because } d^* \text{ is a fixed point of } F
\end{aligned}$$

As for the pairs in β_{k+1}, the statement holds as required by the theorem, and again by the inductive assumption the same holds for pairs in δ_{k+1}. Hence, the statement of the theorem is proven. ■

If the above lemma describes the relationship to the limiting metric, we introduce the second requirement on $\hat{h}_k$ that will allow to relate our framework to the sequence of metrics described in the previous section. For this, we will need to define two important quantities:

$$\begin{aligned}
\#(k, s, s') &:= |\{k' \leq k : (s, s') \in \alpha_{k'}\}| \\
\eta(k) &:= \min_{(s,s')} \#(k, s, s')
\end{aligned}$$

The first quantity keeps track of the number of *exact* updates that have been performed on every pair. The second quantity, $\eta(k)$, contains the number of updates that is guaranteed for all pairs in Ω_S. A last quantity will be defined under the assumption that an update schedule is chosen such that the distances between all pairs of states are updated infinitely often as we increase the total number of updates. For a fixed integer m, we define k_m to be

$$k_m = \min_k \{\eta(k) \geq m\}$$

That is, k_m is the earliest time in the schedule when the guaranteed number of updates over the entire space Ω_S is m. Under these definitions, the following lemma can be proven:

Lemma 4. *Let $m > 0$ and assume that $\eta(k) \to \infty$ as $k \to \infty$. Further assume that $\hat{h}_k \geq h_k$. Then, for any integer m, $F^m(0) \leq h_{k_m}$.*

Proof: We will prove the statement by induction on m. The statement clearly holds for $m = 0$ since both values are equal to 0:

$$k_0 = 0 \Rightarrow h_{k_0} = h_0 = 0 \qquad F^0(0) = 0$$

Now, assume $F^m(0) \leq h_{k_m}$, and we proceed to study changes between consecutive values of

$$k \in \{k_m + j \mid 0 \leq j \leq k_{m+1} - k_m\}$$

For these values of k, we prove, by nested induction on j, that

$$F^{m+1}(0)(s, s') \leq h_k(s, s') \quad \forall (s, s') \in \bigcup_{k'=k_m+1}^{k} \alpha_{k'}$$

In the base case for the nested induction (i.e., $j = 0$), the above statement clearly applies since the set for which we want the property to hold is empty.

Now, for any other l in the desired interval, assume that the statement is true for all $0 \leq j' < l - k_m$. Let $(s, s') \in \alpha_l$. We would like to apply the monotonicity of F in Lemma 2, but for this we need to analyse h_{l-1}. There are a few cases that have to be considered for the choice of partitions (α, β, δ) in previous steps (i.e., $j < l - k + m$).

If $(\bar{s}, \bar{s}') \in \bigcup_{l'=k_m+1}^{l-1} \alpha_{l'}$, then

$$\begin{aligned}
h_{l-1}(\hat{s}, \hat{s}') &\geq F^{m+1}(0)(\bar{s}, \bar{s}') && \text{by hypothesis of} \\
& && \quad \text{the nested-induction} \\
&\geq F^m(0)(\bar{s}, \bar{s}') && \text{by monotonicity of } F
\end{aligned}$$

If $(\bar{s}, \bar{s}') \notin \bigcup_{l'=k_m+1}^{l} \alpha_{l'}$, then for every j' such that $(\bar{s}, \bar{s}') \in \delta_{j'}$, the following holds

$$h_{j'}(\bar{s}, \bar{s}') \geq h_{j'-1}(\bar{s}, \bar{s}')$$

and for every j' such that $(\bar{s}, \bar{s}') \in \beta_{j'}$,

$$h_{j'}(\bar{s}, \bar{s}') = h_{j'-1}(\bar{s}, \bar{s}')$$

By a simple inductive argument, it must hold that

$$\begin{aligned} h_{l-1}(\bar{s}, \bar{s}') &\geq h_{k_m}(\bar{s}, \bar{s}') \\ &\geq F^m(0)(\bar{s}, \bar{s}') \quad \text{by hypothesis of} \\ &\qquad\qquad\qquad\qquad \text{the induction on } m \end{aligned}$$

This concludes the fact that $h_{l-1} \geq F^m(0)$. We now go back to the pair (s, s') from α_l.

$$\begin{aligned} h_l(s, s') &= F(h_{l-1})(s, s') \quad \text{by definition} \\ &\geq F(F^m(0))(s, s') \quad \text{by monotonicity of } F \\ &\qquad\qquad\qquad\qquad \text{on the induction hypothesis} \\ &= F^{m+1}(0)(s, s') \end{aligned}$$

Note that the case when $(s, s') \in \bigcup_{l'=k_m+1}^{l-1} \alpha_{l'} \setminus \alpha_l$ is simpler to analyze, since the value either increases relative to h_{l-1}, or it does not change. Therefore, the assumption of the nested-induction is carried over, and the nested-induction proof is done.

At the same time, by definition of k_m,

$$\bigcup_{k'=k_m+1}^{k_{m+1}} \alpha_{k'} = \Omega_S$$

So that the statement proved with the nested induction can be used to declare that

$$F^{m+1}(0)(s, s') \leq h_{k_{m+1}}(s, s') \quad \forall (s, s') \in \Omega_S.$$

Therefore, $F^{m+1} \leq h_{k_{m+1}}$, which completes the induction on m. ∎

The following Theorem describes the main theoretical result of our work: on-the-fly methods of bisimulation computation can be used to generate a sequence of functions (not necessarily metrics) which converge to the same bisimulation metric.

Theorem 2. *Assume that all pairs of states are picked infinitely often for an update, based on F (i.e., $\eta(k) \to \infty$ as $k \to \infty$). Under the further assumptions of Lemmas 3 and 4, $h_k \to d^*$.*

Proof: By Lemma 3 and Lemma 4, under the definitions given already,

$$F^m(0) \leq h_{k_m} \leq d^*$$

As $k \to \infty$, the number of updates on every state goes to ∞, so $m \to \infty$, and h_{k_m} becomes bounded above and below by d^*. This concludes the proof. ∎

VI. New strategies to compute bisimulation metrics

Theorem 2 opens the door to various computational improvements towards computing bisimulation metrics for MDPs. In this section, we describe different choices for the partitions (α, β, δ) of Ω_S, inspired by asynchronous dynamic programming methods.

First, note that the framework we developed in the previous section is indeed a strict generalization of the original computational scheme presented in [5], which we can emulate using the simplest partitioning strategy:

$$(\alpha_k = \Omega_S, \beta_k = \Omega_S \setminus \alpha_k) \quad \text{for all } k$$

We will refer to this choice as the **"all pairs exact update"** scheme.

The simplest modification to the existing framework is inspired by the Gauss-Seidel method for linear systems, as presented in Section II. The recursive update in 5 illustrates the distinction which is persistent in the "all pairs exact update rule" between h and $F(h)$. We remove this by working with a function h which only gets updated "on-the-fly": we choose an ordering $(s_1, s_1'), (s_2, s_2'), ...$ on Ω_S and cycle the update through Ω_S. Hence, we choose the following partition:

$$(\alpha_k = \{(s_j, s_j')\}, \beta_k = \Omega_S \setminus \alpha_k) \quad \text{for } k = i|\Omega_S| + j$$

We will call this approach the **"Gauss-Seidel update"**.

An alternative to a fixed ordering on the set Ω_S would be to use any random sequence that is guaranteed to select every pair infinitely often. For example, we will study in the next section the **"uniform asynchronous update"**, which picks states using the uniform distribution $U(\Omega_S)$ assigning equal probability to all states.

$$(\alpha_k = \{(s, s')\}, \beta_k = \Omega_S \setminus \alpha_k) \quad (s, s') \text{ picked from } U(\Omega_S)$$

While all the strategies described so far can be used within our theoretical framework, they are oblivious to any partial information from the current distance metric estimate h_k; such information is not used to select the pairs to be updated. Using this information has the potential to lead to more significant gains. Hence, we propose to use an approach similar to prioritized sweeping: changes at a pair (s, s') should be propagated to pairs that are influenced by this update. From the definition of the Monge-Kantorovich transportation problem we can deduce that in Equation (7), $\pi_{ij} = 0$ for all i, j such that $P(i) = 0$ or $Q(j) = 0$. It then becomes clear that $\mathcal{T}(h, P_{sa}, P_{s'a})$ depends only on $h(\bar{s}, \bar{s}')$ such that $P^s_{\bar{s}a} > 0$ for some a, and $P^s_{\bar{s}'a'} > 0$ for some a', and Equation (5) is only relevant for these pairs. Under the computational framework presented in the previous section, the above information can be used to design an informed update rule, which we'll denote by **"prioritized sweeping update"** (the recursive definition is using a queue Q of pairs from Ω_S):

1) For $k = 1, 2, ...$
   a) If queue is empty, initialize Q with a pair picked from $U(\Omega_S)$

b) Assign $\alpha_k = \{(s, s')\}$, where the (s, s') is the top of Q (and remove it from Q) and $\beta_k = \Omega_S \setminus \alpha_k$
c) $\Delta = h_k(s, s') - h_{k-1}(s, s')$
d) Add all pairs $(\bar{s}, \bar{s}')$ to the queue with priority

$$\Delta \sum\nolimits_a \left(P^s_{\bar{s}a} + P^{s'}_{\bar{s}'a} \right)$$

only if both $P^s_{\bar{s}a}, P^{s'}_{\bar{s}'a} > 0$; if the pair is already in the queue, increase its priority to the above.

The priority is designed as follows: one should be eager to update the distance information of a pair $(\bar{s}, \bar{s}')$ if the cost between states that are important in its transportation problem has increased substantially.

For the prioritized sweeping update, the fact that pairs are picked infinitely often follows from the property $h_k \leq d^*$ of Lemma 3, and as a consequence, $\Delta \to 0$ and the priority of all states goes to 0 as well.

VII. Empirical results

In this section we present an illustrative example of the proposed approach, and we show the empirical advantage of on-the-fly bisimulation updates. We use the standard grid-world domain, which has been used extensively in artificial intelligence, as a model for navigation tasks in environments with obstacles. This domain is simple and intuitive, and at the same time, it provides a convenient testbed for comparisons based on the exact bisimulation metric. Grid-world MDPs have sparse transition maps: under any state-action pair, the next state is stochastically chosen from a relatively small subset of S. This suits well the purpose of the current presentation, which is to show how different scheduling for bisimulation updates can lead to better performance, independent of the transportation problem (whose complexity is directly related to the size of state neighbourhood).

As illustrated in Figure 1, we work with stochastic grid-worlds, where reward is received only upon entering a given goal state; all other transitions are neither rewarded, nor penalized. However, using a discount factor $\gamma < 1$ means that an optimal policy has to reach the goal state in minimum expected time. The actions are directional navigation moves (North, South, East and West). All actions are stochastic: with probability 0.9, an action moves the system to the next state in the desired direction, while with the remaining probability 0.1 , the movement is a uniform choice in any of the other directions allowed (see Figure 2 for a visual representation of this effect). Also, if a corresponding transition results in collision with a wall, then the state does not change. The reward is 1 when the system moves into the goal state, and 0 everywhere else.

First, we investigate the behaviour of the strategies discussed in Section VI in the smaller 9×11 MDP, where all methods are expected to converge quickly to the exact bisimulation metric. The standard algorithm, "all pairs update", is run as an iterative method, where each iteration consists of $|S| \times (|S| - 1)/2$ updates, one for every distinct pair. The "Gauss-Seidel update" is almost identical in structure. For the other algorithms, we grouped $|S| \times (|S| - 1)/2$ consecutive updates and we have analyzed the quality of the metric after each group was processed(i.e. one iteration).

Figure 3 presents a summary of the results; we measured the L_1 distance (sum of absolute difference in each entry) between the approximation and the exact bisimulation metric, either as a function of the number of iterations performed, or as a function of the clock running time of the algorithm. Note that the environment is small enough to allow one to compute the exact bisimulation metrics; in large domains, this would not be the case. In such cases, one can plot the sum of all distances (which is always increasing with more iterations) in order to track the progress of the algorithm. Both graphs show essentially the same behaviour. The uniform heuristic performs worst, as expected; this shows that not all "on-the-fly" algorithms are expected to help, and care must be put into selecting the update schedule. The "Gauss-Seidel" strategy provides a slight improvement over the "all-pairs algorithm". The "prioritized sweeping" strategy yields a significant improvement over the previous methods, showing that it is worthwhile to focus on state pairs where the distance change is expected to be largest.

We repeated the same experimental procedure on a larger MDP, of size 30×30. The results are presented in Figure 4. Note that for this larger grid we did not plot the uniform strategy, since its performance was significantly worse than of the other methods. As before, prioritized sweeping generates a good approximation to the bisimulation metric in significantly fewer iterations, by focusing on important pairs of states. However, in terms of computational time, at least using our implementation, this advantage did not materialize in a substantial reduction. Surprisingly, the "Gauss-Seidel" strategy was virtually tied with the prioritized sweeping in terms of running time, despite requiring more iterations. The discrepancy between the performance in terms of number of iterations vs the running time is due to the fact that prioritized sweeping requires a priority queue of the next state pairs that should be processed. Hence, the quality of the queue implementation becomes a factor in larger environments, where the queue grows. We did not yet consider improvements to this aspect of the algorithm, but this is clearly worth pursuing. It is important to note that both "Gauss-Seidel" and "prioritized sweeping" converged faster that the "all-pairs algorithm", both in terms of number of iterations and running time. In particular, "Gauss-Seidel" was comparatively better in the large environment than in the small environment. We expect that in even bigger domains, such effects would be more pronounced. These effects show that there can indeed be a significant benefit from on-the-fly bisimulation metric evaluation.

VIII. Conclusion and Future Work

We presented new theoretical results that establish the foundation of on-the-fly bisimulation metric algorithms, leading to significant computational gains compared to previous approaches. We presented the theoretical development in the context of Markov Decision Proceses. However, all the theory

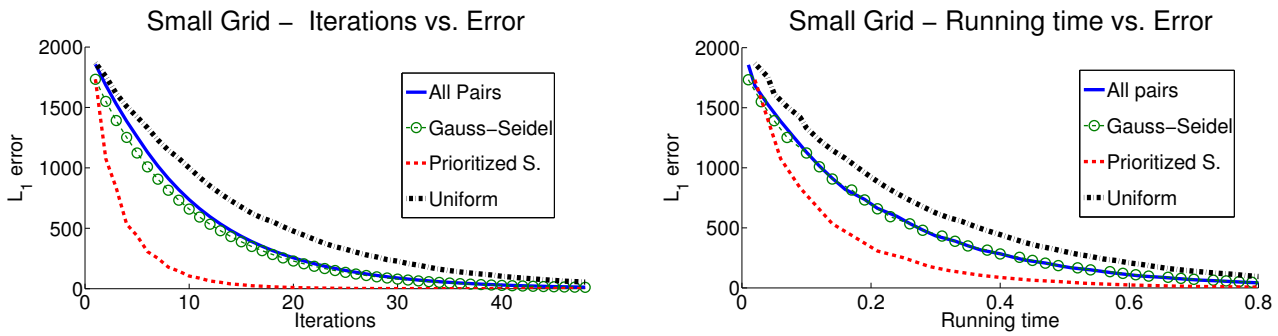

Fig. 3. *9×11-Grid World*: Approximation error is graphed both in terms of the running time of each method, as well as the number of iterations(i.e. updates) performed

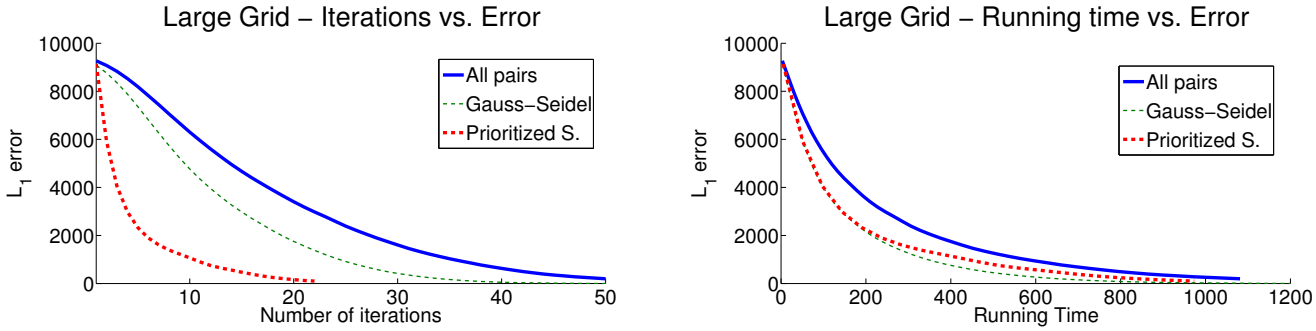

Fig. 4. *30×30-Grid World*: The two best performing strategies are compared in a larger MDP. The difference between the two plots illustrates the fact that the computational cost of maintaining an informed update schedule has to be controlled

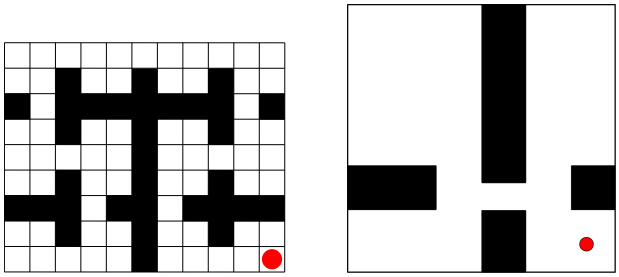

Fig. 1. Small MDP of size 9×11 on the left, and larger MDP of size 30×30 on the right (for visual clarity, only the walls are depicted for the large task, instead of all the cells). Reward is received upon entering the state at the position indicated by the red dot.

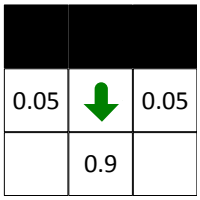

Fig. 2. Stochasticity of transitions in a grid (action in this case is South)

and empirical ideas can also be applied readily to bisimulation metrics for Labelled Markov Processes.

We note that the computational savings depend on properties of the system under scrutiny, such as the possible number of successors from a given state. Such factors need to be taken into account when deciding on the approach to be used. Domain-dependent heuristics can be very useful to manage well the transportation problems that need to be solved. Our theoretical results allow many such approaches to be used, but more empirical experience with different approaches is needed, especially for models of real systems.

One of the open implementation issues is the management of the priority queue for the prioritized sweeping heuristic. We obtained the results presented here by using a standard C++ library implementation. As seen in our results, even though the number of updates with this approach may be significantly smaller, the cost of maintaining the queue can become large. Limiting the size of the queue or using a branch-and-bound approach, in which only a fixed number of high-priority predecessors are enqueued, could potentially improve the running time a lot. We will explore such solutions in future work.

One algorithmic idea that we are currently pursuing is to work with subsets of states, rather than individual states, and split these gradually during the metric computation. This co-inductive splitting idea matches very well the prioritized sweeping approach, and promises to bring significant further computational gains.

This paper tackled finite-state environments. Bisimulation metrics have been defined for continuous domains as well [4], and their computation in this case is based on similar transportation problems. It would be worthwhile to pursue an extension of the theory presented here to the continuous case.

Prioritized sweeping and asynchronous dynamic programming have already been used successfully in robotic environments with continuous states, so this avenue is very promising.

ACKNOWLEDGMENTS

To be added after review.

REFERENCES

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

[2] R. Givan, T. Dean, and M. Greig, "Equivalence notions and model minimization in Markov decision processes," *Artificial Intelligence*, vol. 147, no. 1-2, pp. 163–223, 2003.

[3] 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.

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

[5] N. Ferns, P. Panangaden, and D. Precup, "Metrics for finite Markov decision precesses," in *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, July 2004, pp. 162–169.

[6] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 1994.

[7] L. Kaelbling and M. Littman, "Reinforcement learning: a survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[8] P. S. Castro and D. Precup, "Using bisimulation for policy transfer in mdps," in *Proceedings of the 24th AAAI Conference on Artificial Intelligence*. Association for the Advancement of Artificial Intelligence, 2010.

[9] G. Comanici and D. Precup, "Basis function discovery using spectral clustering and bisimulation metrics," in *AAAI*, 2011.

[10] P. Castro and D. Precup, "Automatic construction of temporally extended actions for mdps using bisimulation metrics," in *Proceedings of the 9th European Workshop on Reinforcement Learning*, 2011.

[11] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains : A survey," *Journal of Machine Learning Research*, vol. 10, pp. 1633–1685, 2009.

[12] J. B. Orlin, "A Faster Strongly Polynomial Minimum Cost Flow Algorithm," in *ACM Symposium on Theory of Computing*. ACM Press, Mar. 1988, pp. 377–387.

[13] J. Fernandez, L. Mounier, C. Jard, and T. Jeron, "On-the-fly verification of finite transition systems," *Formal Methods in System Design*, vol. 1, no. 2-3, pp. 251–273, 1992.

[14] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT Press, 2008.

[15] R. Iosif and R. Sisto, *SPIN Model Checking and Software Verification*, ser. Lecture Notes In Computer Science. Springer-Verlag, 2000, vol. 1885, ch. Using garbage collection in model checking.

[16] D. Bertsekas, "Distributed dynamic programming," *IEEE Transactions on Automatic Control*, pp. 610–616, 1982.

[17] D.P.Bertsekas, "Distributed asynchronous computation of fixed points," *Mathematical Programming*, pp. 107–120, 1983.

[18] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[19] C. Villani, *Topics in optimal transportation*, ser. Graduate Studies in Mathematics. American Mathematical Society, 2003, vol. 58.

[20] S. T. Rachev, *Probability Metrics and the Stability of Stochastic Models*. J. Wiley, 1991.

[21] O. Pele and M. Werman, "Fast and robust earth mover's distances," in *Proceedings of the IEEE International Conference on Computer Vision*, 2009, pp. 460–467.

[22] P. Panangaden, *Labelled Markov Processes*. Imperial College Press, 2009.

[23] L. A. Hageman and D. M. Young, *Applied Iterative Methods*. Academic Press, 1981.

[24] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation*. Prentice-Hall, 1989.

[25] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[26] A. W. Moore and C. G. Atkeson, "Prioritized sweeping: Reinforcement learning with less data and less real time," *Machine Learning*, vol. 13, pp. 103–130, 1993.

[27] S. Chernova and M. Veloso, "Confidence-based policy learning from demonstration using gaussian mixture models categories and subject descriptors," in *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, 2007.

[28] D. Ferguson and A. Stentz, "Focussed processing of MDPs for path planning," in *Proceedings of the IEEE International Conference on Tools with Artificial Intelligence*. IEEE Press, 2004, pp. 310–317.

[29] L. C. Evans and W. Gangbo, "Differential Equations Methods for the MongeKantorovich Mass Transfer Problem," *Mem. Amer. Math. Soc.*, vol. 137, no. 653, pp. viii+66, 1999.

[30] M. L. Fredman and R. E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," *J. ACM*, vol. 34, no. 3, pp. 596–615, Jul. 1987.

[31] Y. Rubner, C. Tomasi, and L. J. Guibas, "The Earth Movers distance as a metric for image retrieval," *International Journal of Computer Vision*, vol. 40, no. 2, pp. 99–121, 2000.

[32] H. Ling and K. Okada, "An efficient Earth Mover's Distance algorithm for robust histogram comparison." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 5, pp. 840–853, 2007.