computational cost becomes a function of m only. In particular, the cost of solving $\bar{M}$ through dynamic programming becomes polynomial in m instead of n: while one application of $\hat{T}$, the Bellman operator of $\hat{M}$, is $O(n\hat{n}|A|)$, the computation of $\bar{T}$ is $O(m^2|A|)$. Therefore, KBSF's time and memory complexities are only linear in n.

We note that, in practice, KBSF's computational requirements can be reduced even further if one enforces the kernels κ_τ^a and $\bar{\kappa}_{\bar{\tau}}$ to be sparse. In particular, given a fixed $\bar{s}_i$, instead of computing $\bar{\mathrm{k}}_{\bar{\tau}}(\bar{s}_i, s_j^a)$ for $j = 1, 2, ..., n_a$, one can evaluate the kernel on a pre-specified neighborhood of $\bar{s}_i$ only. Assuming that $\bar{\mathrm{k}}_{\bar{\tau}}(\bar{s}_i, s_j^a)$ is zero for all s_j^a outside this region, one avoids not only computing the kernel but also storing the resulting values (the same reasoning applies to the computation of $\mathrm{k}_\tau(\hat{s}_i^a, \bar{s}_j)$ for a fixed $\hat{s}_i^a$).

4.1 A closer look at KBSF's approximation

As outlined in Section , KBRL defines the probability of a transition from state $\hat{s}_i^b$ to state $\hat{s}_k^a$ as being $\kappa_\tau^a(\hat{s}_i^b, s_k^a)$, where $a, b \in A$ (see Figure 2a). Note that the kernel κ_τ^a is computed with the initial state s_k^a, and not $\hat{s}_k^a$ itself. The intuition behind this is simple: since we know the transition $s_k^a \xrightarrow{a} \hat{s}_k^a$ has occurred before, the more "similar" $\hat{s}_i^b$ is to s_k^a, the more likely the transition $\hat{s}_i^b \xrightarrow{a} \hat{s}_k^a$ becomes (Ormoneit and Sen, 2002).

From (13), it is clear that the computation of matrices $\mathbf{K}^a$ performed by KBSF follows the same reasoning underlying the computation of KBRL's matrices $\hat{\mathbf{P}}^a$; in particular, $\kappa_\tau^a(\bar{s}_j, s_k^a)$ gives the probability of a transition from $\bar{s}_j$ to $\hat{s}_k^a$. However, when we look at matrix $\mathbf{D}$ things are slightly different: here, the probability of a "transition" from $\hat{s}_i^b$ to representative state $\bar{s}_j$ is given by $\bar{\kappa}_{\bar{\tau}}(\hat{s}_i^b, \bar{s}_j)$—a computation that involves $\bar{s}_j$ itself. If we were to strictly adhere to KBRL's logic when computing the transition probabilities to the representative states $\bar{s}_j$, the probability of transitioning from $\hat{s}_i^b$ to $\bar{s}_j$ upon executing action a should be a function of $\hat{s}_i^b$ and a state s' from which we knew a transition $s' \xrightarrow{a} \bar{s}_j$ had occurred. In this case we would end up with one matrix $\mathbf{D}^a$ for each action $a \in A$. Note though that this formulation of the method is not practical, because the computation of the matrices $\mathbf{D}^a$ would require a transition $(\cdot) \xrightarrow{a} \bar{s}_j$ for each $a \in A$ and each $\bar{s}_j \in \bar{S}$. Clearly, such a requirement is hard to fulfill even if we have a generative model available to generate sample transitions.

In this section we provide an interpretation of the approximation computed by KBSF that supports our definition of matrix $\mathbf{D}$. We start by looking at how KBRL constructs the matrices $\hat{\mathbf{P}}^a$. As shown in Figure 2a, for each action $a \in A$ the state $\hat{s}_i^b$ has an associated stochastic vector $\hat{\mathbf{p}}_j^a \in \mathbb{R}^{1\times n}$ whose nonzero entries correspond to the kernel $\kappa_\tau^a(\hat{s}_i^b, \cdot)$ evaluated at $s_k^a, k = 1, 2, \ldots, n_a$. Since we are dealing with a continuous state space, it is possible to compute an analogous vector for any $s \in \mathbb{S}$ and any $a \in A$. Focusing on the nonzero entries of $\hat{\mathbf{p}}_j^a$, we define the function

$$\begin{aligned} \hat{\mathcal{P}}_{S^a} :\ & \mathbb{S} \mapsto \mathbb{R}^{1\times n_a} \\ & \hat{\mathcal{P}}_{S^a}(s) = \hat{\mathbf{p}}^a \iff \hat{p}_i^a = \kappa_\tau^a(s, s_i^a) \text{ for } i = 1, 2, ..., n_a. \end{aligned} \tag{16}$$

Clearly, full knowledge of the function $\hat{\mathcal{P}}_{S^a}$ allows for an exact computation of KBRL's transition matrix $\hat{\mathbf{P}}^a$. Now suppose we do not know $\hat{\mathcal{P}}_{S^a}$ and we want to compute an