Modular Policy Composition with Policy Centroids

Sandesh Adhikary
Paul G. Allen School of Computer Science
University of Washington
adhikary@cs.washington.edu

Byron Boots
Paul G. Allen School of Computer Science
University of Washington
bboots@cs.washington.edu

Abstract

We consider the task of aggregating policies in multi-objective decision making problems where multiple policies are trained to accomplish potentially conflicting tasks. While policy composition is a crucial component of multi-objective problems, commonly used techniques are restricted to simple averages that do not adhere to or exploit the structure of policy spaces. We present a new framework for policy composition viewing the problem as computing centroids in distance spaces where policies are embedded. These *policy centroids* not only subsume various existing composition techniques, but also provide a new means of inducing useful properties in composite policies through judicious choices of embedding spaces and distances. Additionally, we introduce policy centroids that extend existing compositions to new problem settings. For deterministic policies, we use distances between state-action value functions to define utility centroids that can be tuned to adjust the intensity of individual policy preferences. For stochastic policies, we introduce policy centroids based on statistical distances including the maximum mean discrepancy, which are particularly useful when policies are accessible only through samples. We evaluate our proposed policy centroids on various illustrative examples that highlight their benefits over existing approaches.

Keywords: policy composition, multi objective reinforcement learning, modular reinforcement learning

1 Introduction

As we continue to expand the scale and complexity of automated decision making, it becomes increasingly difficult to quantify all desired objectives into a single monolithic goal. We are routinely faced with multiple competing objectives with no concrete notion of a composite goal. Autonomous vehicles must quickly reach their destinations while avoiding crashes; economic policy should bolster growth but also minimize inequality; financial investments need to maximize returns while maintaining a diverse portfolio. Even when a composite objective does exist, it may still be difficult to learn a policy to accomplish it. Various works in multi-objective reinforcement learning have demonstrated the benefit of training separate agents on decompositions of complex reward signals. In robotics, a complex robot can be decomposed into multiple sub-agents, each reacting to changes in its local environment [1]. Even when tackling a single objective, we can benefit from training an ensemble of policies using different algorithms [2].

The problem of modular policy composition has been tackled in various fields, with numerous proposed solutions including sums and products of experts [2], compositions of utilities or potential fields [3], voting based ranking, and so on. The central idea behind these compositions is to form an average policy from a set of recommendations. However, existing approaches tend to use fairly limited notions of averages, generally restricted to arithmetic means, geometric means, and extremums. We consider the task of averaging policies in its full generality as computing centroids over arbitrary distance spaces that minimize distances to individual policies. This perspective can subsume many existing compositions, and also serve as a principled framework to design new compositions by picking appropriate embedding spaces and distances. We consider three existing approaches to policy composition, and show how policy centroids not only generalize them, but can also extend them to new problem settings.

2 Policy Centroids

We tackle policy composition as finding a single-policy solution to multi-objective Markov decision processes (MOMDPs). A MOMDP is a tuple $(\mathcal{S},\mathcal{A},T,\gamma,R)$ of state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $T : S \times \mathcal{A} \times \mathcal{A} \to [0,1]$, discount factor $\gamma \in [0,1]$, and a *vector-valued* reward function $R : S \times A \times S \to \mathbb{R}^T$. The components of the rewards vector r_i are the rewards for the T separate objectives. A policy π determines the action executed at a given state; a deterministic policy $\pi(s) : \mathcal{S} \to \mathcal{A}$ returns a single action, and a stochastic policy $\pi(a|s) : \mathcal{A} \to [0,1]$ is a distribution over $\mathcal{A}$. For a state-action pair (s,a), $Q_i(a|s) = \mathbb{E}_{\pi_i}(\sum_{t=0}^{\infty} \gamma^t([r_i]_t)|s_0 = s, a_0 = a)$ denotes the state-action value function under the policy π_i for the i-th objective.

Policy Centroids Consider a set $\{\pi_i\}_{i=1}^T$ of T policies, such that each policy π_i has been formulated separately based only on the i-th component of the reward function. These policies must now be combined into a single composite policy $\bar{\pi}$, without assuming access to a composite reward function or a known decomposition of the composite objective into individual objectives. Let (M_i,d_i) be distance spaces[1] where π_i can be compared with alternatives via distances d_i. Policies are mapped onto these spaces using policy embeddings $\mu_i(\pi)$. Finally, given a set of optional non-negative weights $\{w_i\}_{i=1}^T$ with $\sum_i w_i = 1$, the composite policy $\bar{\pi}$ or the *policy centroid* is defined as follows

$$\bar{\pi} = \operatorname*{argmin}_{\pi} \sum_i w_i d_i(\mu_i(\pi_i), \mu_i(\pi)). \tag{1}$$

To design a policy centroid, we pick (1) embeddings μ_i that capture important features of policies, and (2) distances d_i that account for meaningful differences between them. The weights w_i are linear relative importances of the T objectives. Such weights are used in many existing policy compositions as the primary mode of controlling the behavior of the composite policy. These weights are either set using knowledge of the composite objective or learned with respect to a composite reward; otherwise, they are set to be uniformly distributed. Policy centroids provide additional means of affecting properties of the composite policy through the choice of the distance spaces (M_i,d_i). In the following sections, we show how various existing compositions can be interpreted as policy centroids. Moreover, we provide examples where different choices of embeddings μ_i and distances d_i can lead to composite policies with useful properties, and extend existing compositions to new problem settings.

2.1 Utility Centroids

In value-based reinforcement learning (RL), we often have access Q-values $Q(a|s)$ serving as utility maps over actions $a \in \mathcal{A}$ for a given state s, which are natural choices for policy embeddings. Unsurprisingly, averaging Q-values is already a popular composition strategy known as *utility fusion* [3], where the arithmetic average of Q-values is treated as the composite Q-value. We introduce *utility centroids* that subsume utility fusion. Given an exponent $\beta > 0$, the utility centroid[2] is defined as

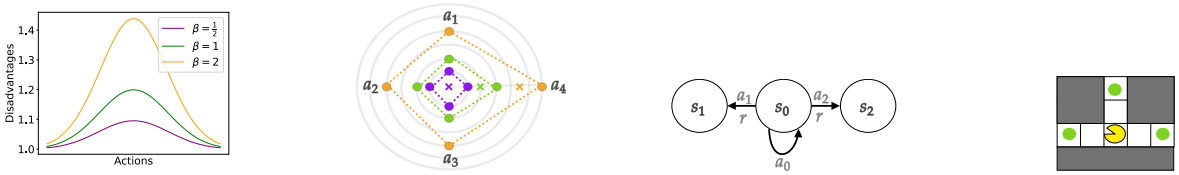
$$\bar{a} = \operatorname*{argmin}_{a} \sum_i w_i (Q_i(a_i|s) - Q_i(a|s) + 1)^{\beta}. \tag{2}$$

Here, task-specific embeddings are simply the Q-values $\mu_i(a|s) = Q_i(a|s)$ and distances are $d_i(Q_i(a_1|s), Q_i(a_2|s)) = (|Q_i(a_1|s) - Q_i(a_2|s)| + 1)^{\beta} - 1$. Equation 2 minimizes the average disadvantage or opportunity cost $(Q_i(a_i|s) - Q_i(a|s) + 1)^{\beta}$ of picking a over the preferred $a_i = \operatorname{argmax}_a Q_i(a|s)$. As illustrated in Figure 1a, the exponent β controls the intensity of individual preferences. Setting $\beta > 1$ (resp. $\beta < 1$) magnifies (resp. minimizes) differences in disadvantages leading to sharper (resp. flatter) disadvantage curves.

The exponent β plays a similar role to weights w_i; while w_i stretch or squeeze distances between policies linearly, β magnifies or minimizes distances between policies exponentially. However, unlike weights w_i, we can set the same exponent β across all policies and still affect relative distances. In Figure 1b, filled-circles of the same color correspond to the recommended

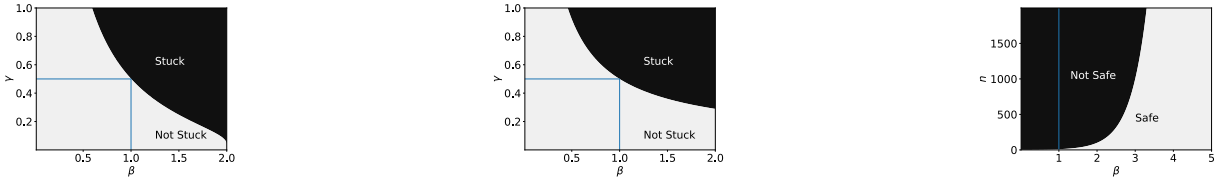
[1] A distance space (M,d) is a set M equipped with a non-negative, symmetric, and reflexive distance function $d(x,x')$ for $x,x' \in M$.

[2] The optional offset +1 simply sets the domain of the exponential to $\mathbb{R}_{\geq 1}$, avoiding the inverse behavior of the exponent for $(0,1)$.



(a) (b) (c) (d)

Figure 1: (a-b) Varying β in utility centroids in Equation 2 (c-d) Illustrations of MDPs with attractor states.



(a) Attractors: 3 state MDP (b) Attractors: PacBoy (c) Tyranny of the majority

Figure 2: Utility centroids to prevent the tyranny of the majority and escape attractors. The blue lines are reference markers for utility fusion, or equivalently utility centroids ($\beta=1$)

actions from four policies for a given choice of β. When $\beta=1$ (green), the policy recommending action a_4 has a slightly lower disadvantage, pushing the centroid closer to a_4. With $\beta>1$ (orange) differences in disadvantages are magnified, pushing the centroid further towards a_4. With $\beta<1$ (purple), the small difference in disadvantages is minimized, and the centroid is not biased towards any one policies. With $\beta=1$, Equation 2 reduces to standard utility fusion. Thus, β can serve as a useful hyperparameter, especially in the absence of non-uniform weights w_i. As we now discuss, adjusting β can alleviate some known problems with utility fusion in such settings.

Preventing Attractors When the Q-values of multiple policies are trained *egocentrically* (without taking into account the composite Q-value), utility fusion with uniform weights w_i overestimates the value of states where policies disagree on the optimal action [4]. For large discount factors γ, these states can become attractors where the agent gets stuck; restricting us to myopic suboptimal policies with low discount factors. In Equation 2, reducing β essentially minimizes disagreement between policies and can help us escape attractors. To demonstrate, we consider the two examples from [4] illustrating attractors in utility fusion. In the first example (Figure 1c), we consider a 3 state deterministic MOMDP where the agent in state s_0 can stay in place with action a_0, or move to one of two terminating goal states s_1 and s_2 with actions a_1 and a_2 respectively. We define separate objectives with reward $r>0$ for reaching s_1 and s_2 respectively, and no reward for staying in place. In the second example in Figure 1d, the PacBoy agent is equidistant from a reward in either of the three directions $\{\uparrow,\leftarrow,\rightarrow\}$ in a simple deterministic MOMDP; choosing $\downarrow$ keeps it in place. In both examples, egocentric utility fusion with uniform weights w_i gets stuck when $\gamma>0.5$ [4]. In Figures 2a and 2b, we plot the combinations of γ and β for which the policy *centroid* in Equation 2 is stuck (shaded black) and for which it reaches one of the two goal states (shaded white). As we decrease β, we can allow larger discount factors while still avoiding getting stuck. However, using too small a β may also lead to undesirable behavior where the relative preferences of policies are ignored.

Preventing the tyranny of the majority Utility fusion can suffer from a tyranny of the majority, where an objective dominated by a large number of competing objectives may never be fulfilled. If the minority objective is safety critical, it may be desirable to forego the other goals to prevent an accident. If the objectives involve human stakeholders, this majority bias can be unfair – the minority stakeholder may lose confidence in the system if their objectives are consistently ignored in service of the majority. As an illustration, consider a set of T objectives $\{O_i\}_{i=1}^{T}$ defined over a MOMDP with two actions $\{a_1,a_2\}$; O_1 corresponds to avoiding accidents, while the others correspond to reaching various goals. At some state, O_1 identifies action a_2 as unsafe with a disadvantage of d. In contrast, a_2 is slightly favorable over a_1 for the goal-seeking objectives, assigning disadvantage $\epsilon<d$ for a_1. With only two tasks $\{O_1,O_2\}$, utility fusion opts for the safe action a_1 as it has lower disadvantage. However, as we add other objectives to the mix, the unsafe action a_2 is picked when $n>d/\epsilon$. With the utility centroid in Equation 2, we can increase β to intensify individual preferences, leading to a larger threshold. In Figure 2c, we plot various combinations of β and n for which the agent picks the unsafe action a_2 (shaded black) and the safe action a_1 (shaded white) when $\epsilon=0.1d$, i.e., each goal-seeking agent only favors the unsafe action by a small margin of 10%. However, as β becomes very large, the minority may exert too much influence, preventing the agent from ever achieving its other goals. Thus β will likely have to be tuned to balance these effects.

2.2 Stochastic Policy Centroids

We now turn to the problem of composing policies represented as probability distributions $\pi(a|s)$ over actions, which we refer to as stochastic policies. Such policies can arise, for instance, due to exploration or as solutions to partially observable MDPs or adversarial games. Two common approaches of composing a set of stochastic policies $\{\pi_i\}$ are through sums and products of experts:

$$\bar{a}=h(\bar{\pi}(a|s)) \qquad \text{Sum of Experts (SoE):} \quad \bar{\pi}\propto\sum_i \pi_i(a|s) \qquad \text{Product of Experts (PoE):} \quad \bar{\pi}\propto\prod_i \pi_i(a|s). \tag{3}$$

The function $h(\cdot)$ could be an argmax operation over the composite density $\bar{\pi}$, or a function drawing samples from it. While not always described as such, these compositions are equivalent to element-wise arithmetic and geometric means, which are special cases of generalized f-means, which, in turn, are instances of Fréchet centroids. We can thus formulate policy centroids

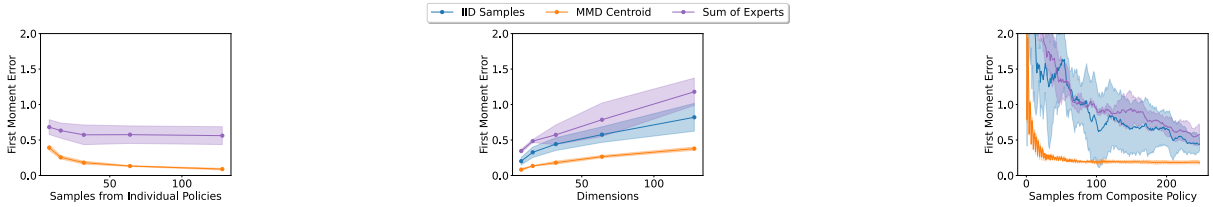


Figure 3: Errors for MMD centroids and SoE composition as a function of (left) samples from individual policies (middle) dimensionality of $\mathcal{A}$ and (right) samples from the composite policy. Error bars are standard deviations across 3 random seeds

corresponding to generalized-means-of-experts (GMoE) through a monotonic, continuous and invertible function f as follows:

$$\bar{\pi} \propto \operatorname*{argmin}_{\pi} \sum_i d^2(\pi_i, \pi) \quad \text{where} \quad d^2(\pi, \pi') = \sum_a |f(\pi(a|s)) - f(\pi'(a|s))|^2. \tag{4}$$

This centroid defines the well-known class of generalized f-means. With $f(x) = x^\beta$ (for $\beta \neq 0$), we get the family of *power means*, which includes SoE (arithmetic mean for $\beta = 1$) and PoE (geometric mean for $\beta \to 0$), as well as a spectrum of other means.

Composing Implicit Stochastic Policies While GMoEs are intuitive and ubiquitous, they may not always be ideal, for instance, when composing *implicit* stochastic policies. Implicit policies $\hat{\pi}$ are distributions without a known density function, but allow sampling. Such policies can be useful, for instance, to incorporate unknown noise distributions or as a flexible policy class for complex action distributions [5]. Given sets of actions $\{a_j^{(i)} \sim \hat{\pi}_i\}$ drawn from a set of implicit policies $\{\hat{\pi}_i\}$, we now wish to generate new actions $\{\bar{a}_m\}$ from the composite policy $\bar{\pi}$. As a concrete example, consider the following SoE composition problem where we wish to evaluate the expected value of some function $g(a)$ under the distribution of the composite policy. Assuming that both the function g and individual policies are computationally expensive to query, we seek a good estimate of $\mathbb{E}_{\bar{\pi}}\, g(a)$ with minimal samples from individual policies as well as the composite policy. SoEs are not directly applicable since they require the policies' probability densities. We would thus need some form of density estimation, which can be sample intensive for high-dimensional actions. With policy centroids, we can extend SoEs to this new problem setting. Specifically, we can embed implicit policies onto reproducing kernel Hilbert spaces, and use the maximum mean discrepency between distributions as our distance function.

MMD Policy Centroids The maximum mean discrepancy (MMD) is an integral probability (pseudo) metric (IPM) between probability measures that captures the maximal difference between expectations of functions. For a set of functions $\mathcal{G}$, the IPM between distributions π and π' is the supremum of the difference in expectations $\mathbb{E}_\pi(g) - \mathbb{E}_{\pi'}(g)$ over all $g \in \mathcal{G}$. We obtain the MMD when $\mathcal{G}$ is the unit-ball in a reproducing kernel Hilbert space (RKHS) [6]. For any symmetric positive-definite kernel $k(\cdot,\cdot)$ on a domain $\mathcal{A} \times \mathcal{A}$, there exists a corresponding RKHS $\mathcal{H}_k$ and k is the canonical feature mapping to $\mathcal{H}_k$. Additionally, we can use the kernel to also map probability distributions π defined over $\mathcal{A}$. These embedded distributions $\mu_\pi \in \mathcal{H}_k$, known as *kernel mean embeddings*, are defined as $\mu_\pi = \int k(a, \cdot) d\pi(a)$. The empirical estimates of these embeddings $\hat{\mu}_\pi = \frac{1}{n} \sum_{j=1}^n k(a_j \sim \pi, \cdot)$ converge in MMD to μ_π at a rate of $O(n^{-1/2})$ [6], which is independent of the dimensionality of $\mathcal{A}$ (in contrast to the curse of dimensionality for density estimation). The MMD between two kernel mean embeddings is the $\mathcal{H}_k$-norm of their difference: $\text{MMD}(\hat{\pi}, \hat{\pi}') = || \hat{\mu}_\pi - \hat{\mu}_{\pi'} ||_{\mathcal{H}_k}$ Additionally, if the kernel is *characteristic* (e.g. Gaussian, Laplace), the MMD is a metric such that $\text{MMD}(\pi, \pi') = 0 \Leftrightarrow \pi = \pi'$ [6].

Given a kernel k, we define *MMD policy centroids* as $\bar{a} \sim \bar{\pi} = \operatorname{argmin}_{\hat{\pi}} \sum_i w_i \text{MMD}(\hat{\pi}_i, \hat{\pi}) = \operatorname{argmin}_{\hat{\mu}} \sum_i w_i ||\hat{\mu}_i - \hat{\mu}||^2_{\mathcal{H}_k}$. This objective is essentially what is optimized by the kernel herding algorithm [7] that draws samples from a kernel mean embedding; here, the target is instead the weighted sum $\sum_i w_i \hat{\mu}_i$. Using kernel herding, we can draw the t-th sample $\bar{a}_t$ from $\bar{\pi}$ as follows

$$\bar{a}_t = \operatorname*{argmin}_a \sum_{i,j} -2 w_i k(a_j^{(i)}, a) + I(t > 1) \sum_{m=1}^{t-1} \frac{1}{t} k(\bar{a}_m, a) \qquad \text{where } I(t > 1) \text{ is the indicator function.} \tag{5}$$

While the first term in the objective encourages $\bar{a}_t$ to be similar to the set of samples $\{a_j^{(i)} \sim \hat{\pi}_i\}$, the second term encourages it to be *dissimilar* to samples $\{\bar{a}_m\}_{m=1}^{t-1}$ generated in prior iterations. This latter repulsive force improves sample diversity by inducing negative autocorrelations. For bounded kernels corresponding to finite dimensional $\mathcal{H}_k$, the herded *super* samples converge in MMD at a rate of $O(n^{-1})$ to the target distribution – faster than $O(n^{-1/2})$ for iid sampling from the true distribution [7]. Even though $\mathcal{H}_k$ is generally not finite, we still tend to get fast convergence in practice – even if Equation 5 is only solved approximately [7]. Thus, MMD centroids are well-suited for our purposes. Firstly, using the herded super samples, we should require fewer evaluations of g before $\mathbb{E}_{\bar{\pi}}(g)$ converges. Secondly, since we bypass an explicit density estimation, we should need fewer samples from individual policies as well. Moreover, since the convergence rate of the *implicit* density estimation $\hat{\mu} \approx \mu$ is independent of the dimensions of $\mathcal{A}$, MMD centroids should scale better to high-dimensional action spaces.

To demonstrate the advantages of MMD centroids, we consider a set of 3 Gaussian policies $\{\pi_i = \mathcal{N}(a_i, 0.1\mathbb{I})\}$ with mean actions $a_i \in \mathbb{R}^d$. Assuming uniform policy weights w_i, the composite distribution for the true weighted SoE policy is simply $\sum_i \frac{1}{3} \mathcal{N}(a_i, 0.1\mathbb{I})$. Our goal is to generate M samples $\{\bar{a}_m\}_{m=1}^M$ from the composite policy (only using samples from π_i) and use them to compute expected values of functions. Here, we fix the target function to be the first moment (mean action) of the composite policy, which is simply $\frac{1}{3} a_i$. In Figure 3, we compare estimation errors for traditional SoE, MMD centroids, as well as iid samples from the true composite policy. For MMD centroids, we use the Gaussian kernel with its bandwidth set using the median-trick heuristic, and generate samples via kernel herding. Since the true composite policy is generally unknown, we avoid tuning hyperparameters for the optimization in Equation 5 by opting for a brute force search within a random subset of M action samples drawn from individual policies. For the SoE composition (Equation 3), we draw samples from the kernel density estimate of the composite policy us-

ing a Gaussian kernel (with the median-trick bandwidth). In Figure 3, we fix $(d=32, M=250)$, $(M=250, N=32)$ and $(d=32, N=32)$ respectively, and plot errors with respect to varying N (samples from individual policies), d (dimensionality of actions), and M (samples from the composite policy). We see that MMD centroids offered better scaling with respect to all three parameters.

2.3 Riemannian Motion Policies

We have thus far focused on two policy compositions often encountered in multi-objective RL. We now turn our attention to a different setting within the framework of Riemannian motion policies (RMPs) from robotics, designed to combine multiple acceleration based policies for cohesive motion generation [1]. As described below, policy centroids share the intuition behind RMP-composition in using task-specific distance spaces, and can also extend RMP-composition to the setting of stochastic controllers.

RMP Composition While a robot's complete state is defined on a configuration space $\mathcal{Q}$, its various objectives can be defined on separate task spaces $\mathcal{X}_i$. Configuration space states $\mathbf{q}$ are mapped onto task-space states $\mathbf{x}_i$ via task-maps $\phi_i : \mathcal{Q} \to \mathcal{X}_i$. Using $\mathbf{J}_i$, the Jacobians of the task-maps ϕ_i, we can also map configuration space velocities and accelerations into their task-space counterparts as $\dot{\mathbf{x}}_i = \mathbf{J}_i \dot{\mathbf{q}}$ and $\ddot{\mathbf{x}}_i = \mathbf{J}_i \ddot{\mathbf{q}} + \dot{\mathbf{J}}_i \dot{\mathbf{q}}$. A task-space RMP over $\mathcal{X}_i$ is a tuple $(\mathbf{f}_i, \mathbf{M}_i)$ indicating a recommended force $\mathbf{f}_i = \mathbf{M}_i \ddot{\mathbf{x}}_i$ given a PSD inertia matrix $\mathbf{M}_i$. The general idea behind the RMP framework is to combine task-space RMPs defining desired accelerations $\ddot{\mathbf{x}}_i$ into a composite RMP in the configuration space. Given a set of task-space RMPs $\{(\mathbf{f}_i, \mathbf{M}_i)\}$, the composite configuration-space RMP $(\mathbf{f}, \mathbf{M})$ is computed by *pulling-back* the task-space forces into the configuration space as $\texttt{pullback}(\mathbf{f}_i) = \mathbf{J}_i^T (\mathbf{f}_i - \mathbf{M}_i \dot{\mathbf{J}}_i \dot{\mathbf{q}}_i)$ and $\texttt{pullback}(\mathbf{M}_i) = \mathbf{J}_i^T \mathbf{M}_i \mathbf{J}_i$, and then adding up the pulled-back forces $\mathbf{f} = \sum_i^T \texttt{pullback}(\mathbf{f}_i)$ and inertia matrices $\mathbf{M} = \sum_i \texttt{pullback}(\mathbf{M}_i)$. The composite control or acceleration $\ddot{\mathbf{q}} = \mathbf{M}^+ \mathbf{f}$ is the solution to the following objective function

$$\operatorname*{argmin}_{\ddot{\mathbf{q}}} \frac{1}{2} \sum_{i=1}^{T} \| \mathbf{J}_i \ddot{\mathbf{q}} + \dot{\mathbf{J}}_i \dot{\mathbf{q}} - \ddot{\mathbf{x}}_i \|^2_{\mathbf{M}_i} = \operatorname*{argmin}_{\ddot{\mathbf{q}}} \frac{1}{2} \sum_{i=1}^{T} \| \mu_i(\ddot{\mathbf{q}}) - \mu_i(\ddot{\mathbf{x}}_i) \|^2_{\mathbf{M}_i}. \tag{6}$$

In the second equality above, we have rewritten the RMP composition objective similar to a policy centroid with $\mu_i(\ddot{\mathbf{q}}) = \mathbf{J}_i \ddot{\mathbf{q}} + \dot{\mathbf{J}}_i \dot{\mathbf{q}}$, where $\mu_i(\ddot{\mathbf{x}}_i) = \ddot{\mathbf{x}}_i$. Each term in this objective can be interpreted as a task-specific distance (defined via $\mathbf{M}_i$) between $\mu_i(\ddot{\mathbf{q}})$ and desired task-controls $\ddot{\mathbf{x}}_i$. Thus, policy centroids share RMP's underlying idea of employing task-specific mappings and computing centroids with respect to task-specific distances. However, policy centroids extend this idea beyond acceleration-based controls to generic settings including, for instance, discrete action spaces. Moreover, as we now discuss, the policy centroid perspective allows us to extend RMP composition to the case of stochastic controllers.

Wasserstein Policy Centroids The RMP framework assumes deterministic controls $\ddot{\mathbf{x}}_i$ or $\mathbf{f}_i$ and is not directly applicable when these controls are stochastic. Incorporating stochasticity into RMPs can be useful, for instance, when individual controllers are learned via RL with exploration, or to escape local stationary points where competing controls cancel each other out. To compose such stochastic RMP controllers, we can collect samples of desired controls from each individual RMP and pull them back onto the configuration space. As in Equation 6, the composite metric $\mathbf{M}$ defines a distance measure for pulled-back acceleration controls. In the deterministic setting with only one sample per RMP, we would essentially obtain the composite control formed via the metric-weighted averaging in Equation 6. But in the stochastic setting, we need to perform an average both in the space of controls (as in deterministic RMPs), but also in terms of probability masses spread across controls. Essentially, we require some form of stochastic averaging that can take into account the ground distance between controls in the pulled-back space, while also performing a probabilistic average. The family of Wasserstein distances between distributions is exactly such a statistical distance. The p-th Wasserstein distance $\mathcal{W}_p(\pi, \pi'; d)$ between two distributions π and π' defined with respect to a ground metric $d(\cdot, \cdot)$ represents the cost of transforming π into π' by moving probability mass in the ground space of samples; the cost of moving probability mass is measured using d. Using a Wasserstein distance as our statistical distance between policies, we can define Wasserstein policy centroids for RMPs with stochastic controllers as the Wasserstein barycenter of the individual pulled-back control distributions. Similar to how RMP composition computes sums or metric-weighted averages of controls, the Wasserstein policy centroids would allow us to average entire distributions over controls.

3 Conclusion

We presented a framework for modular policy composition, viewing the problem as computing centroids over distance spaces. We showed how this framework not only subsumes various existing policy compositions, but also provided examples of how it can be used to adapt them to new problem settings. In future work, we aim to implement the policy centroids presented here in large-scale policy composition problems, and also integrate policy centroids into policy learning pipelines.

References

[1] C.-A. Cheng, M. Mukadam, J. Issac, S. Birchfield, D. Fox, B. Boots, and N. Ratliff, "Rmpflow: A geometric framework for generation of multitask motion policies," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 3, pp. 968–987, 2021.

[2] M. A. Wiering and H. Van Hasselt, "Ensemble algorithms in reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 4, pp. 930–936, 2008.

[3] J. K. Rosenblatt, "Optimal selection of uncertain actions by maximizing expected utility," *Autonomous Robots*, 2000.

[4] R. Laroche, M. Fatemi, J. Romoff, and H. van Seijen, "Multi-advisor reinforcement learning," *arXiv:1704.00756*, 2017.

[5] Y. Tang and S. Agrawal, "Implicit policy for reinforcement learning," *arXiv preprint arXiv:1806.06798*, 2018.

[6] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *JMLR*, vol. 13, no. 1.

[7] Y. Chen, M. Welling, and A. Smola, "Super-samples from kernel herding," in *UAI*, 2010.