Trajectory Diversity for Zero-Shot Coordination
Supplemental Material

A Additional Intuition and Method Details

In this section we give additional intuition behind our method and provide the full derivations to the results stated in the paper.

A.1 Trajectory view

As opposed to the state-centric view used in a majority of RL works, TrajeDi relies upon a trajectory-centric view, which we detail and formalize here.

As stated in section 3, the core of our objective formulation is the set $\mathcal{T}$ of all possible trajectories. This set naturally forms a tree with two types of nodes: actions and states. The children of a state node s are all the possible actions in that state, while the children of an action node a are all the states s' that satisfy $P(s'|s,a) > 0$. The root is the starting state s_0, and leaves correspond to all final states s_T, duplicated by the number of possible trajectories leading to that state. As such, there is a one-to-one correspondence between trajectories and the leaves of the tree. In settings with randomized initial states, it suffices to add a place-holder root x, of which the children are the possible initial states and the construction still holds.

If the behaviour of a policy π is determined by the distribution $\pi(\tau)$ it produces over trajectories, then we can equivalently characterize it in terms of the probability of reaching each tree leaf while following π.

This tree view also casts new light on the effect of γ in eq. 5. When $\gamma = 1$, the $\delta_{i,t}(\tau)$ weigh all trajectory actions equally. Therefore, TrajeDi acts globally and "only" pushes policies to reach disjoint subsets of all leaves. In particular, there will be no pressure for two policies to act differently in parts of the tree that one of them has extremely low probability of ever reaching. For the same reason, two policies can have arbitrarily long sequences of identical actions on a given rollout, provided they branch out at any point before the end so as to reach different leaves. In this case, reducing γ produces a more local $\delta_{i,t}(\tau)$, thus requiring policies to select different actions more frequently along any rollout.

Effectively, this trajectory tree is a game tree akin to those used in AI programs for two-player perfect information games Knuth & Moore (1975), with one "player" being the policy and the other being the environment itself in our case. Also, ours is not the first formulation of POMDPs as a trajectory tree. Indeed, Kearns et al. present a similar construction, but restrict themselves exclusively to state nodes and therefore cannot represent environments with stochastic transitions. As a result, our formulation is a strict generalization of theirs, with the two becoming equivalent in deterministic settings.

A.2 JSD$_0$ as the KL divergence on individual actions

Setting $\gamma = 0$ in TrajeDi makes $\delta_{i,t}(\tau) = \pi_i(a_t^\tau|s_t^\tau)$ and $\hat{\delta}_t(\tau) = \tilde{\pi}(a_t^\tau|s_t^\tau)$, which produces

$$\text{JSD}_0(\pi_1, ..., \pi_n) := -\frac{1}{n}\sum_{i=1}^{n}\sum_{\tau}\mathbb{P}(\tau|\pi_i)\sum_{t=0}^{T}\frac{1}{T}\log\frac{\tilde{\pi}(a_t^\tau|s_t^\tau)}{\pi_i(a_t^\tau|s_t^\tau)}. \tag{A.1}$$

Rearranging the terms, and changing the summation to be over all partial trajectories τ_t and actions a_t, we see that

$$\begin{aligned}\mathrm{JSD}_0(\pi_1, ..., \pi_n) &= -\frac{1}{nT}\sum_{i=1}^{n}\sum_{t=0}^{T}\sum_{\tau_t}\mathbb{P}(\tau_t|\pi_i)\sum_{a\in\mathcal{A}}\pi_i(a|s_t^\tau)\log\frac{\tilde{\pi}(a|s_t^\tau)}{\pi_i(a|s_t^\tau)}\\ &= \frac{1}{nT}\sum_{i=1}^{n}\sum_{t=0}^{T}\sum_{\tau_t}\mathbb{P}(\tau_t|\pi_i)\mathcal{D}_{KL}\Big(\pi_i(\cdot|s_t^\tau)\Big|\Big|\tilde{\pi}(\cdot|s_t^\tau)\Big)\end{aligned}$$

and so JSD_0 measures the KL divergence from the mean at the level of individual actions, in a state-wise manner. Maximizing JSD_0 therefore produces policies that differ in their action choices at every state. Albeit close, this is technically different from the action-level JSD in that the KL divergence for each π_i is assigned a weight $\mathbb{P}(\tau_t|\pi_i)$, which is different across policies. Nonetheless, it remains bounded in $[0, \log n]$.

A.3 Exact JSD gradient

Here we derive an exact gradient for the TrajeDi objective

$$\mathrm{JSD}_\gamma = -\frac{1}{n}\sum_{j=1}^{n}\sum_{\tau}\varepsilon(\tau)\frac{1}{T}\sum_{t=0}^{T}\pi_j(\tau)\underbrace{\left[\log\frac{1}{n} + \log\left(n\hat{\delta}_t(\tau)\right) - \log\delta_{j,t}(\tau)\right]}_{f(\tau,j,t)}.$$

Taking the gradient with respect to policy parameters $\boldsymbol{\theta}_i$ yields

$$\nabla_{\boldsymbol{\theta}_i}\mathrm{JSD}_\gamma = -\frac{1}{n}\sum_{\tau}\varepsilon(\tau)\frac{1}{T}\sum_{t=0}^{T}\sum_{j=1}^{n}\left[\nabla_{\boldsymbol{\theta}_i}\pi_j(\tau) * f(\tau,j,t) + \pi_j(\tau) * \nabla_{\boldsymbol{\theta}_i}f(\tau,j,t)\right],$$

where

$$\nabla_{\boldsymbol{\theta}_i}\pi_j(\tau) = \begin{cases}0 & \text{if } i\neq j\\ \pi_i(\tau)\nabla_{\boldsymbol{\theta}_i}\log\pi_i(\tau) & \text{if } i = j\end{cases}$$

$$\nabla_{\boldsymbol{\theta}_i}f(\tau,j,t) = \begin{cases}\nabla_{\boldsymbol{\theta}_i}\log\left(n\hat{\delta}_t(\tau)\right) = \frac{n\nabla_{\boldsymbol{\theta}_i}\hat{\delta}_t(\tau)}{n\hat{\delta}_t(\tau)} = \frac{\nabla_{\boldsymbol{\theta}_i}\delta_{i,t}(\tau)}{n\hat{\delta}_t(\tau)} = \frac{\delta_{i,t}(\tau)}{n\hat{\delta}_t(\tau)}\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau) & \text{if } i\neq j\\ \nabla_{\boldsymbol{\theta}_i}\log\left(n\hat{\delta}_t(\tau)\right) - \nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau) = \left(\frac{\delta_{i,t}(\tau)}{n\hat{\delta}_t(\tau)} - 1\right)\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau) & \text{if } i = j.\end{cases}$$

Plugging this back into the equation above, we obtain:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}_i}\mathrm{JSD}_\gamma = &-\frac{1}{n}\sum_{\tau}\varepsilon(\tau)\frac{1}{T}\sum_{t=0}^{T}\Bigg[\ \pi_i(\tau)\nabla_{\boldsymbol{\theta}_i}\log\pi_i(\tau)f(\tau,i,t) + \pi_i(\tau)\left(\frac{\delta_{i,t}(\tau)}{n\hat{\delta}_t(\tau)} - 1\right)\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau)\\ &+ \sum_{j=1,j\neq i}^{n}\pi_j(\tau)\frac{\delta_{i,t}(\tau)}{n\hat{\delta}_t(\tau)}\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau)\Bigg]\\ &= -\frac{1}{n}\sum_{\tau}\varepsilon(\tau)\frac{1}{T}\sum_{t=0}^{T}\Bigg[\ \pi_i(\tau)\nabla_{\boldsymbol{\theta}_i}\log\pi_i(\tau)f(\tau,i,t) - \pi_i(\tau)\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau)\\ &+ \sum_{j=1}^{n}\pi_j(\tau)\frac{\delta_{i,t}(\tau)}{n\hat{\delta}_t(\tau)}\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau)\Bigg]\end{aligned}$$

We now expand $f(\tau, \cdot, t)$ and rearrange the terms:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}_i} \mathrm{JSD}_\gamma &= -\frac{1}{n}\sum_\tau \varepsilon(\tau)\frac{1}{T}\sum_{t=0}^{T}\Bigg[\ \pi_i(\tau)\nabla_{\boldsymbol{\theta}_i}\log\pi_i(\tau)\left[\log\frac{1}{n}+\log\left(n\hat{\delta}_t(\tau)\right)-\log\delta_{i,t}(\tau)\right] \\
&\quad -\pi_i(\tau)\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau)+\sum_{j=1}^{n}\pi_j(\tau)\frac{\delta_{i,t}(\tau)}{n\hat{\delta}_t(\tau)}\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau)\Bigg] \\
&= -\mathbb{E}_{\tau\sim\pi_i}\left\{\frac{1}{nT}\sum_{t=0}^{T}\log\left(\frac{\hat{\delta}_t(\tau)}{\delta_{i,t}(\tau)}\right)\nabla_{\boldsymbol{\theta}_i}\log\pi_i(\tau)-\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau)\right\} \\
&\quad -\frac{1}{n}\sum_{j=1}^{n}\mathbb{E}_{\tau\sim\pi_j}\left\{\frac{1}{T}\sum_{t=0}^{T}\frac{\delta_{i,t}(\tau)}{n\hat{\delta}_t(\tau)}\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau)\right\} \\
\nabla_{\boldsymbol{\theta}_i} \mathrm{JSD}_\gamma &= -\mathbb{E}_{\tau\sim\pi_i}\left\{\frac{1}{nT}\sum_{t=0}^{T}\log\left(\frac{\hat{\delta}_t(\tau)}{\delta_{i,t}(\tau)}\right)\nabla_{\boldsymbol{\theta}_i}\log\pi_i(\tau)\right\} \\
&\quad -\mathbb{E}_{\pi}\mathbb{E}_{\tau\sim\pi}\left\{\frac{1}{T}\sum_{t=0}^{T}\frac{\delta_{i,t}(\tau)}{n\hat{\delta}_t(\tau)}\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau)\right\}.
\end{aligned}$$

In the last step here, we use the fact that $\mathbb{E}_{\tau\sim\pi_i}\left\{\frac{1}{T}\sum_{t=0}^{T}\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau)\right\}=0$, which we prove in the next section.

To use TrajeDi in off-policy settings, we must distinguish between the policy being updated, π_i and the one that generated the trajectories, μ_i, which is the sampling policy. To do so, it suffices to multiply by $\frac{\mu_i}{\mu_i}$ and extract importance sampling ratios, leaving us with:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}_i} \mathrm{JSD}_\gamma = -\mathbb{E}_{\tau\sim\mu_i}\Bigg\{&\frac{1}{nT}\sum_{t=0}^{T}\frac{\pi_i(\tau)}{\mu_i(\tau)}\left(\log\hat{\delta}_t(\tau)-\log\delta_{i,t}(\tau)\right)\nabla_{\boldsymbol{\theta}_i}\log\pi_i(\tau) \\
&+\frac{\hat{\pi}(\tau)\delta_{i,t}(\tau)}{\mu_i(\tau)\hat{\delta}_t(\tau)}\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau)\Bigg\}.
\end{aligned}$$

Evaluating the TrajeDi exact gradient, even when sampling only a subset of trajectories, carries a computational cost that is quadratic in the size of the population. This is because the $\hat{\delta}_t(\tau)$ term inside the logarithm requires that each policy be unrolled on the trajectories generated by all of the other policies. This is prohibitively expensive in many cases and so motivates our approximate objective and gradient, which allow sampling a subset of all population to unroll for every gradient update.

A.4 Approximate JSD objective and Gradient

Here we provide the full derivation for the gradient estimator presented in eq. 6. In doing so, we also show a different variant of the gradient which has lower variance but requires training every policy on trajectories produced by the other policies in the population.

We first expand eq. 5 and approximate the third term in the sum by its tangent. Letting the arguments $(\pi_1, ..., \pi_n)$ be implied, we thus obtain (note the usage of index j instead of i):

$$\begin{aligned}
\mathrm{JSD}_\gamma &= -\frac{1}{n}\sum_{j=1}^{n}\sum_\tau \varepsilon(\tau)\pi_j(\tau)\frac{1}{T}\sum_{t=0}^{T}\left[\log\frac{1}{n}+\log\left(n\hat{\delta}_t(\tau)\right)-\log\delta_{j,t}(\tau)\right] \\
\tilde{\mathrm{JSD}}_\gamma &= -\frac{1}{n}\sum_{j=1}^{n}\sum_\tau \varepsilon(\tau)\frac{1}{T}\sum_{t=0}^{T}\pi_j(\tau)\underbrace{\left[\log\frac{1}{n}+n\hat{\delta}_t(\tau)-1-\log\delta_{j,t}(\tau)\right]}_{f(\tau,j,t)},
\end{aligned}$$

with $\tilde{\text{JSD}}_\gamma \leq \text{JSD}_\gamma$.

We now take the gradient with respect to $\boldsymbol{\theta}_i$, getting

$$\nabla_{\boldsymbol{\theta}_i} \tilde{\text{JSD}}_\gamma = -\frac{1}{n}\sum_\tau \varepsilon(\tau)\frac{1}{T}\sum_{t=0}^{T}\sum_{j=1}^{n}\left[\nabla_{\boldsymbol{\theta}_i}\pi_j(\tau) * f(\tau,j,t) + \pi_j(\tau) * \nabla_{\boldsymbol{\theta}_i} f(\tau,j,t)\right],$$

where

$$\nabla_{\boldsymbol{\theta}_i}\pi_j(\tau) = \begin{cases} 0 & \text{if } i \neq j \\ \pi_i(\tau)\nabla_{\boldsymbol{\theta}_i}\log\pi_i(\tau) & \text{if } i = j \end{cases}$$

$$\nabla_{\boldsymbol{\theta}_i} f(\tau,j,t) = \begin{cases} n\nabla_{\boldsymbol{\theta}_i}\hat{\delta}_t(\tau) = n\nabla_{\boldsymbol{\theta}_i}\sum_{k=1}^n \frac{1}{n}\delta_{k,t}(\tau) = \nabla_{\boldsymbol{\theta}_i}\delta_{i,t}(\tau) = \delta_{i,t}(\tau)\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau) & \text{if } i \neq j \\ \nabla_{\boldsymbol{\theta}_i}\delta_{i,t}(\tau) - \nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau) = (\delta_{i,t}(\tau)-1)\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau) & \text{if } i = j. \end{cases}$$

Plugging this back in, we obtain:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}_i} \tilde{\text{JSD}}_\gamma = &-\frac{1}{n}\sum_\tau \varepsilon(\tau)\frac{1}{T}\sum_{t=0}^{T}\Bigg[\ \pi_i(\tau)\nabla_{\boldsymbol{\theta}_i}\log\pi_i(\tau) f(\tau,i,t) + \pi_i(\tau)(\delta_{i,t}(\tau)-1)\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau) \\ &+ \sum_{j=1, j\neq i}^{n}\pi_j(\tau)\delta_{i,t}(\tau)\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau)\Bigg] \\ &= -\frac{1}{n}\sum_\tau \varepsilon(\tau)\frac{1}{T}\sum_{t=0}^{T}\Bigg[\ \pi_i(\tau)\nabla_{\boldsymbol{\theta}_i}\log\pi_i(\tau) f(\tau,i,t) - \pi_i(\tau)\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau) \\ &+ \sum_{j=1}^{n}\pi_j(\tau)\delta_{i,t}(\tau)\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau)\Bigg]\end{aligned}$$

Expanding and rearranging further,

$$\begin{aligned}\nabla_{\boldsymbol{\theta}_i} \tilde{\text{JSD}}_\gamma = &-\frac{1}{n}\sum_\tau \varepsilon(\tau)\frac{1}{T}\sum_{t=0}^{T}\Bigg[\pi_i(\tau)\nabla_{\boldsymbol{\theta}_i}\log\pi_i(\tau)\left[\log\frac{1}{n} + n\hat{\delta}_t(\tau) - 1 - \log\delta_{i,t}(\tau)\right] \\ &- \pi_i(\tau)\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau) + \sum_{j=1}^{n}\pi_j(\tau)\delta_{i,t}(\tau)\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau)\Bigg] \\ &= -\mathbb{E}_{\tau\sim\pi_i}\left\{\frac{1}{T}\sum_{t=0}^{T}\left(\hat{\delta}_t(\tau) - \frac{1}{n}\log\delta_{i,t}(\tau)\right)\nabla_{\boldsymbol{\theta}_i}\log\pi_i(\tau) - \frac{1}{n}\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau)\right\} \\ &- \frac{1}{n}\sum_{j=1}^{n}\mathbb{E}_{\tau\sim\pi_j}\left\{\frac{1}{T}\sum_{t=0}^{T}\delta_{i,t}(\tau)\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau)\right\}.\end{aligned}$$

Here, in the second step we remove the factor $\log\frac{1}{n} - 1$ since that term is equal to 0 in expectation. Then, changing the outer sum of the second term into an expectation, we get

$$\begin{aligned}&= -\mathbb{E}_{\tau\sim\pi_i}\left\{\frac{1}{T}\sum_{t=0}^{T}\left(\hat{\delta}_t(\tau) - \frac{1}{n}\log\delta_{i,t}(\tau)\right)\nabla_{\boldsymbol{\theta}_i}\log\pi_i(\tau) - \frac{1}{n}\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau)\right\} \\ &- \mathbb{E}_\pi\mathbb{E}_{\tau\sim\pi}\left\{\frac{1}{T}\sum_{t=0}^{T}\delta_{i,t}(\tau)\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau)\right\}\end{aligned}$$

Finally we show that $\mathbb{E}_{\tau\sim\pi_i}\left\{\frac{1}{T}\sum_{t=0}^{T}\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau)\right\}$ equals 0, which allows to further reduce the variance of the gradient estimator.

$$\begin{aligned}
\mathbb{E}_{\tau\sim\pi_i}\left\{\frac{1}{T}\sum_{t=0}^{T}\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau)\right\} &= \mathbb{E}_{\tau\sim\pi_i}\left\{\frac{1}{T}\sum_{t=0}^{T}\sum_{x=0}^{T}\nabla_{\boldsymbol{\theta}_i}\log\left[\pi_i(a_x^\tau|\tau_x)\right]^{\gamma^{|t-x|}}\right\} \\
&= \mathbb{E}_{\tau\sim\pi_i}\left\{\frac{1}{T}\sum_{x=0}^{T}\underbrace{\left(\sum_{t=0}^{T}\gamma^{|t-x|}\right)}_{c(x)}\nabla_{\boldsymbol{\theta}_i}\log\pi_i(a_x^\tau|\tau_x)\right\} \\
&= \sum_{\tau}\mathbb{P}(\tau|\pi_i)\frac{1}{T}\sum_{x=0}^{T}c(x)\nabla_{\boldsymbol{\theta}_i}\log\pi_i(a_x^\tau|\tau_x) \\
&= \frac{1}{T}\sum_{x=0}^{T}\sum_{\tau}\mathbb{P}(\tau_x|\pi_i)\pi_i(a_x^\tau|\tau_x)\mathbb{P}(\tau_{x+1:T}|\tau_x,a_x^\tau,\pi_i)c(x)\nabla_{\boldsymbol{\theta}_i}\log\pi_i(a_x^\tau|\tau_x).
\end{aligned}$$

In the last step here, we simply expand the trajectory probability into its terms before, at and after reaching τ_x, which is a unique state node in the trajectory tree $\mathcal{T}$. In this view, $\mathbb{P}(\tau_x|\pi_i)$ is the probability of reaching node τ_x, $\pi_i(a_x^\tau|\tau_x)$ is the probability of transitioning from τ_x to its child action node a_x^τ and $\mathbb{P}(\tau_{x+1:T}|\tau_x,a_x^\tau,\pi_i)$ is the probability of staying on the path τ from a_x^τ forward. Then, because many trajectories share the same path segment up to node τ_x, we can factor the corresponding probability terms to change our inner sum into three nested ones:

$$\begin{aligned}
&\frac{1}{T}\sum_{x=0}^{T}\sum_{\tau_x}\sum_{a}\sum_{\tau_{x+1:T}}\mathbb{P}(\tau_x|\pi_i)\pi_i(a|\tau_x)\mathbb{P}(\tau_{x+1:T}|\tau_x,a)c(x)\nabla_{\boldsymbol{\theta}_i}\log\pi_i(a|\tau_x) \\
&= \frac{1}{T}\sum_{x=0}^{T}\sum_{\tau_x}\mathbb{P}(\tau_x|\pi_i)\sum_{a}\pi_i(a|\tau_x)c(x)\nabla_{\boldsymbol{\theta}_i}\log\pi_i(a|\tau_x)\underbrace{\sum_{\tau_{x+1:T}}\mathbb{P}(\tau_{x+1:T}|\tau_x,a)}_{=1} \\
&= \frac{1}{T}\sum_{x=0}^{T}c(x)\sum_{\tau_x}\mathbb{P}(\tau_x|\pi_i)\sum_{a}\pi_i(a|\tau_x)\nabla_{\boldsymbol{\theta}_i}\log\pi_i(a|\tau_x) \\
&= \frac{1}{T}\sum_{x=0}^{T}c(x)\sum_{\tau_x}\mathbb{P}(\tau_x|\pi_i)\mathbb{E}_{a\sim\pi_i(\cdot|\tau_x)}\left\{\nabla_{\boldsymbol{\theta}_i}\log\pi_i(a|\tau_x)\right\} \\
&= \frac{1}{T}\sum_{x=0}^{T}c(x)\sum_{\tau_x}\mathbb{P}(\tau_x|\pi_i)*0=0,
\end{aligned}$$

where we exploit the fact that $\mathbb{E}_{x\sim\mathcal{P}(x)}\{\nabla\log\mathcal{P}(x)\}=0$.

And so, the final gradient estimator is:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}_i}\tilde{\text{JSD}}_\gamma = &-\mathbb{E}_{\tau\sim\pi_i}\left\{\frac{1}{T}\sum_{t=0}^{T}\left(\hat{\delta}_t(\tau)-\frac{1}{n}\log\delta_{i,t}(\tau)\right)\nabla_{\boldsymbol{\theta}_i}\log\pi_i(\tau)\right\} \\
&-\mathbb{E}_{\pi}\mathbb{E}_{\tau\sim\pi}\left\{\frac{1}{T}\sum_{t=0}^{T}\delta_{i,t}(\tau)\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau)\right\}.
\end{aligned} \tag{A.2}$$

Here, the first term is computed on-policy and is reflective of the trajectory sampling procedure induced by π_i, while the second term is computed along trajectories generated by any policy in the population, and can therefore be associated with how much the distribution induced by π_i overlaps with the average one.

Despite giving us more freedom in our choice of which policy to evaluate on which trajectories, the gradient estimator in eq. A.2 still requires caution. Indeed, to estimate the gradients for the entire population, it is necessary that each policy be evaluated on its own trajectories, but also on trajectories selected from at least one other policy at random. This can be avoided by multiplying by $\frac{\pi_i}{\pi_i}$ in the second term and rearranging the equation so as to change the expectation support from $\hat{\pi}$ to π_i. Doing so we obtain

$$\begin{aligned}\nabla_{\boldsymbol{\theta}_i} \tilde{\text{JSD}}_\gamma = -\mathbb{E}_{\tau\sim\pi_i}\Bigg\{ &\frac{1}{T}\sum_{t=0}^{T}\left(\hat{\delta}_t(\tau) - \frac{1}{n}\log\delta_{i,t}(\tau)\right)\nabla_{\boldsymbol{\theta}_i}\log\pi_i(\tau) \\ &+ \frac{\hat{\pi}(\tau)}{\pi_i(\tau)}\delta_{i,t}(\tau)\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau)\Bigg\},\end{aligned} \tag{A.3}$$

matching our result from eq. A.3.

Note that in either form, as is the case for the objective in eq. 5, γ has no effect on the sampling procedure as it is only present within the δ's. As a result, given a pre-sampled set of trajectories complete with their corresponding $\pi(\tau)$'s, the TrajeDi gradient and objectives can be computed efficiently for any sensitivity level without the need to query the policies repeatedly. This is practical from an implementation point of view, but also allows to dynamically adjust γ during training by computing the TrajeDi objective for some value and correcting as needed.

A.5 Adapting the gradient to off-policy methods

The gradient in eq. A.3 allows for great flexibility in sampling the policies used to estimate the "average" policy. However, as it is presented, it assumes the probabilities used to compute the objective are the same as those used for generating the rollouts. This does not hold in off-policy methods, including the batch RL setup we use for our Hanabi experiments, and so we must correct for it. Thus, let $\mu_i(\tau) = \prod_{t=0}^{T-1}\mu_i(a_t|s_t)$ be the probability that policy i sampled trajectory τ at the time of the rollout, which we record in the replay buffer along with the trajectory. This is the on-policy probability, and so it suffices to multiply by $\frac{\mu_i(\tau)}{\mu_i(\tau)}$ to change the support of the expectation and obtain a form of eq. A.3 that is usable off-policy:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}_i} \tilde{\text{JSD}}_\gamma = -\mathbb{E}_{\tau\sim\mu_i}\Bigg\{ &\frac{1}{T}\sum_{t=0}^{T}\frac{\pi_i(\tau)}{\mu_i(\tau)}\left(\hat{\delta}_t(\tau) - \frac{1}{n}\log\delta_{i,t}(\tau)\right)\nabla_{\boldsymbol{\theta}_i}\log\pi_i(\tau) \\ &+ \frac{\hat{\pi}(\tau)}{\mu_i(\tau)}\delta_{i,t}(\tau)\nabla_{\boldsymbol{\theta}_i}\log\delta_{i,t}(\tau)\Bigg\}.\end{aligned} \tag{A.4}$$

Eq. A.4 is exactly the TrajeDi gradient we implement for our Hanabi experiments.

B Hanabi Implementation

For Hanabi, we use a distributed computation framework for both training and simulation. On the simulation side we run 6400 Hanabi environments in parallel, assigning multiple environments per thread for greater efficiency. In each environment we have agents that utilize computationally expensive neural networks GPU calls when selecting actions. By running these calls asynchronously, we can accommodate multiple environments per thread, and multiple threads per experiment, thus fully utilizing the GPU. Inspired by Schaul et al. (2016), we use every environment in a permanent simulation loop, and at the end of an episode we take the entire action observation history, which consists of actions, observations, and rewards, pad it, and add it to a central replay buffer. Based on Kapturowski et al. (2019), we compute the priority of each trajectory as $\xi = 0.9 * max_i\xi_i + 0.1 * \xi$, where ξ is expressed as the TD error per step. From a training perspective, we have a training loop that samples the central prioritized replay buffer and computes updates based on the TD error. The simulation policies are updated based on the up-to-date training policy every 10 gradient steps. We are able to accomplish our training and simulation infrastructure with 30 CPU cores and 2 GPUs per agent.

For the population based training, we rely on a client-server implementation. The central server stores a copy of the policy parameters for the entire population, while each client stores its own relevant policy parameters. Every 50 gradient steps the client updates the server with the updated trained policy's parameters and fetches the relevant set of policy parameters that the training policy requires, for instance when computing the TrajeDi loss.

Finally, for the individual agent architecture and hyperparameters, we use the implementation provided by Hu et al. on GitHub[1].

C Additional Results

Method	Self-Play	Cross-Play
OP	**24.24 ± 0.02**	23.65 ± 0.06
OP pool BR	24.17 ± 0.04	23.66 ± 0.07
OP pool BR (slowed down pool)	24.20 ± 0.02	23.88 ± 0.04
OP pool BR + TrajeDi	24.22 ± 0.01	**24.09 ± 0.02**
OP pool BR + TrajeDi (fixed window kernel)	24.16 ± 0.03	23.97 ± 0.04

Table 1: Self-play and cross-play scores for Other-Play (OP) agents in Hanabi, with the last action hidden. In order, the results are for 1) 5 individual agents, 2) BRs to pools of agents, 3) BR to pools of agents with slower training, 4) BR to pools of TrajeDi-regularized agents and 5) BR to pools of TrajeDi-regularized agents using a fixed window kernel. For all but the first row, we trained 4 BRs, each to their own pool of 3 agents. Using TrajeDi to regularize the agents results in the best cross-play scores, at a little cost in terms of self-play. Intervals correspond to one standard error on the mean.

In this section, we provide additional results on using TrajeDi in Hanabi. In the paper, we reported self-play and cross-play score for three different training setups: individual OP agents, BRs to pools of OP agents, and BRs to pools of TrajeDi-regularized OP agents. In addition, one of our hypotheses was that TrajeDi was beneficial to cross-play scores by slowing down the training of the pool of agents, compared to the BR. This in turn would force the BR agent to learn more grounded strategies for playing with the "delayed" pool. To test our hypothesis, we therefore train BRs to pools for which we deliberately slow down training, by an equivalent amount as if we were using TrajeDi. Finally we test a variant of TrajeDi for which we replace the exponential kernel from eq. 4 by a fixed windowed kernel

$$\delta'_{i,t}(\tau) := \prod_{t'=t-w}^{t+w} \pi_i(a^\tau_{t'}|s^\tau_{t'}), \tag{A.5}$$

where w is a positive integer.

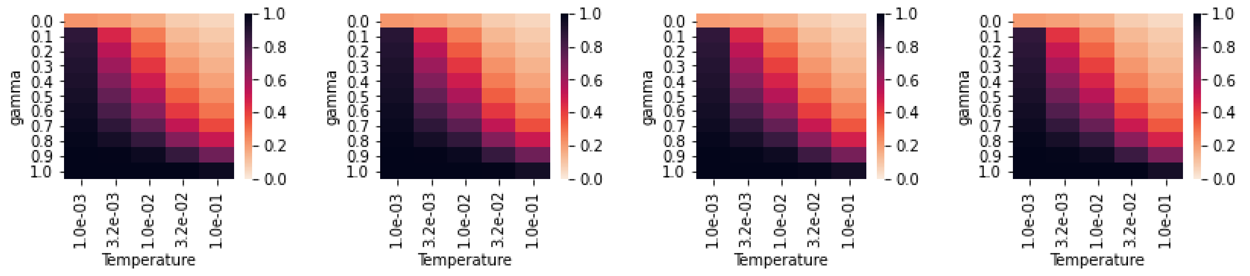
All pools are composed of three agents plus the BR, and we train them to convergence, which takes approximately 48h on our infrastructure. For TrajeDi, we ran a hyperparameter grid search composed of 16 combinations of γ (or w when using the fixed window kernel) and α, and selected the best combination in terms of average self-play score. In the end, the results we report are with $\gamma = 0.5$ and $\alpha = 0.1$ when using the exponential kernel and $w = 2$ and $\alpha = 0.1$ when using the windowed one.

The results (including those already presented in the paper) are summarised in Table 1. As predicted, slowing down the agents in the pool does indeed improve the cross-play scores of the BRs over the baseline. However, this accounts only for about half the cross-play improvement seen with TrajeDi, and so diversity remains necessary for additional gains.

[1]`https://github.com/facebookresearch/hanabi_SAD`

Finally, while replacing the exponential kernel by the windowed one in TrajeDi yields slightly worse results both in self-play and cross-play, the difference between the two variants is not statistically significant. As such, TrajeDi, if properly tuned, remains robust to the shape of the kernel used.

C.1 Measuring Diversity



(a) OP (b) OP (slowed down pool) (c) OP + TrajeDi (d) OP + TrajeDi (window)

Figure 1: TrajeDi score maps of Hanabi pool agents as a function of policy softmax temperature and objective discounting (γ). a) Simple OP pool b) OP pool slowed down relative to its BR c) TrajeDi-regularized OP pool d) TrajeDi-regularized OP pool with windowed kernel.

In addition to self-play and cross-play scores, we also measure the pool diversity for our different settings. Do to so, we evaluate pairwise diversity between the agents of a same pool, and average over all pools with the same settings (but different seeds). For each pair, we generate 50 trajectories per agent, at 5 different temperatures, and compute the TrajeDi score at 11 different values of γ ranging from 0 to 1.

The results are shown in Fig. 1, with a heatmap for each of the 4 pool configurations we tested (simple OP agents, slowed down OP agents, OP regularized with TrajeDi and OP regularized with the windowed kernel variant of TrajeDi). Remark that although here we are only evaluating the diversity of the pools, each of them is still trained with a common BR.

Horizontally, and for all maps, there is a clear monotonic trend for more stochastic policies to have lower scores, due to the increasingly random actions producing higher overlap between trajectory distributions. An equally strong trend can be seen vertically, with higher γ values resulting in higher TrajeDi scores, reflective of the difference between policies on longer time slices. This strongly supports our intuition and our conjecture from the end of section 4.3.

Despite the clear improvement in cross-play scores observed in Table 1 for TrajeDi-regularized training, the diversity heatmaps between pools trained with and without TrajeDi have only very subtle differences. We believe this may be due to TrajeDi having qualitative effects on diversity, such as making it more uniform throughout the rollout rather than concentrating behaviour differences only on the late-game, for instance. We plan on investigating this in the near future.

References

Hu, H., Lerer, A., Peysakhovich, A., and Foerster, J. " other-play" for zero-shot coordination. *arXiv preprint arXiv:2003.02979*, 2020.

Kapturowski, S., Ostrovski, G., Quan, J., Munos, R., and Dabney, W. Recurrent experience replay in distributed reinforcement learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Kearns, M. J., Mansour, Y., and Ng, A. Y. Approximate planning in large pomdps via reusable trajectories. In *Advances in Neural Information Processing Systems*, pp. 1001–1007, 2000.

Knuth, D. E. and Moore, R. W. An analysis of alpha-beta pruning. *Artificial intelligence*, 6(4):293–326, 1975.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. In Bengio, Y. and LeCun, Y. (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL `http://arxiv.org/abs/1511.05952`.