A. Proofs

A.1. Max-Ent Generalized Policy Improvement

Theorem 3.1 (Max-Ent Generalized Policy Improvement) *Let $\pi_1, \pi_2, ..., \pi_n$ be n policies with α-max-ent action-value functions $Q^1, Q^2, ..., Q^n$ and value functions $V^1, V^2, ..., V^n$. Define*

$$\pi(a|s) \propto \exp\left(\tfrac{1}{\alpha} \max_i Q^i(s,a)\right).$$

Then,

$$Q^\pi(s,a) \geq \max_i Q^i(s,a) \text{ for all } s \in \mathcal{S},\ a \in \mathcal{A}, \tag{5}$$

$$V^\pi(s) \geq \max_i V^i(s) \text{ for all } s \in \mathcal{S}, \tag{6}$$

where $Q^\pi(s,a)$ and $V^\pi(s)$ are the α-max-ent action-value and value function respectively of π.

For brevity we denote $Q^{\max} \equiv \max_i Q^i$. Define the soft Bellman operator associated with policy π as

$$\mathcal{T}^\pi Q(s,a) \equiv r(s,a,s') + \gamma \mathbb{E}_{p(s'|s,a)}\left[\alpha H[\pi(\cdot|s')] + \mathbb{E}_{a' \sim \pi(\cdot|s')}\left[Q(s',a')\right]\right].$$

Haarnoja et al. (2018b) have pointed out that the soft Bellman operator $\mathcal{T}^\pi$ corresponds to a conventional, "hard", Bellman operator defined over the same MDP but with reward $r_\pi(s,a,s') = r(s,a,s') + \gamma\alpha\mathbb{E}_{p(s'|s,a)}\left[H[\pi(\cdot|s')]\right]$. Thus, as long as $r(s,a,s')$ and $H[\pi(\cdot|s')]$ are bounded, $\mathcal{T}^\pi$ is a contraction with Q^π as its fixed point. Applying $\mathcal{T}^\pi$ to $Q^{\max}(s,a)$ we have:

$$\begin{aligned}
\mathcal{T}^\pi Q^{\max}(s,a) &= r(s,a,s') + \gamma \mathbb{E}_{s' \sim p(\cdot|s,a), a' \sim \pi(\cdot|s')}\left[-\alpha \log \pi(a'|s') + Q^{\max}(s',a')\right] \\
&= r(s,a,s') + \gamma \mathbb{E}_{s' \sim p(\cdot|s,a), a' \sim \pi(\cdot|s')}\left[-\alpha \log \frac{\exp(\alpha^{-1} Q^{\max}(s',a'))}{Z^\pi(s')} + Q^{\max}(s',a')\right] \\
&= r(s,a,s') + \gamma \mathbb{E}_{s' \sim p(\cdot|s,a)}\left[\alpha \log Z^\pi(s')\right].
\end{aligned}$$

Similarly, if we apply $\mathcal{T}^{\pi_i}$, the soft Bellman operator induced by policy π_i, to $Q^{\max}(s,a)$, we obtain:

$$\mathcal{T}^{\pi_i} Q^{\max}(s,a) = r(s,a,s') + \gamma \mathbb{E}_{s' \sim p(\cdot|s,a), a' \sim \pi_i(\cdot|s')}\left[-\alpha \log \pi_i(a'|s') + Q^{\max}(s',a')\right].$$

We now note that the Kullback-Leibler divergence between π_i and π can be written as

$$\begin{aligned}
D_{\mathrm{KL}}(\pi_i(\cdot|s) \| \pi(\cdot|s)) &= \mathbb{E}_{a \sim \pi_i(\cdot|s)}\left[\log \pi_i(a|s) - \log \pi(a|s)\right] \\
&= \mathbb{E}_{a \sim \pi_i(\cdot|s)}\left[\log \pi_i(a|s) - \frac{1}{\alpha} Q^{\max}(s,a) + \log Z^\pi(s)\right].
\end{aligned}$$

The quantity above, which is always nonnegative, will be useful in the subsequent derivations. Next we write

$$\begin{aligned}
\mathcal{T}^\pi Q^{\max}(s,a) - \mathcal{T}^{\pi_i} Q^{\max}(s,a) &= \gamma \mathbb{E}_{s' \sim p(\cdot|s,a)}\left[\alpha \log Z^\pi(s') - \mathbb{E}_{a' \sim \pi_i(\cdot|s')}[-\alpha \log \pi_i(a'|s') + Q^{\max}(s',a')]\right] \\
&= \gamma \mathbb{E}_{s' \sim p(\cdot|s,a)}\left[\mathbb{E}_{a' \sim \pi_i(\cdot|s')}[\alpha \log Z^\pi(s') + \alpha \log \pi_i(a'|s') - Q^{\max}(s',a')]\right] \\
&= \gamma \mathbb{E}_{s' \sim p(\cdot|s,a)}\left[\alpha D_{\mathrm{KL}}(\pi_i(\cdot|s') \| \pi(\cdot|s'))\right] \\
&\geq 0.
\end{aligned} \tag{14}$$

From (14) we have that

$$\mathcal{T}^\pi Q^{\max}(s,a) \geq \mathcal{T}^{\pi_i} Q^{\max}(s,a) \geq \mathcal{T}^{\pi_i} Q^i(s,a) = Q^i(s,a) \text{ for all } i.$$

Using the contraction and monotonicity of the soft Bellman operator $\mathcal{T}^\pi$ we have

$$Q^\pi(s,a) = \lim_{k \to \infty} (\mathcal{T}^\pi)^k Q^{\max}(s,a) \geq Q^i(s,a) \text{ for all } i.$$

We have just showed (5). In order to show (6), we note that

$$\begin{aligned} V^\pi(s) &\equiv \alpha H[\pi(\cdot|s)] + \mathbb{E}_{a\sim\pi}\left[Q^\pi(s,a)\right] \\ &\geq \alpha H[\pi(\cdot|s)] + \mathbb{E}_{a\sim\pi}\left[Q^{\max}(s,a)\right] \\ &= \alpha \log Z^\pi(s). \end{aligned} \tag{15}$$

Similarly, we have, for all i,

$$\begin{aligned} V^i(s) &= \mathbb{E}_{a\sim\pi_i(\cdot|s)}\left[Q^i(s,a) - \alpha \log \pi_i(a|s)\right] \\ &\leq \mathbb{E}_{a\sim\pi_i(\cdot|s)}\left[Q^{\max}(s,a) - \alpha \log \pi_i(a|s)\right] \\ &= \alpha \log Z^\pi(s) - \alpha D_{\mathrm{KL}}(\pi_i(\cdot|s)\|\pi(\cdot|s)) \\ &\leq \alpha \log Z^\pi(s). \end{aligned} \tag{16}$$

The bound (6) follows from (15) and (16).

A.2. DC Proof

Theorem 3.2 (DC Optimality) *Let π_i, π_j be α max-ent optimal policies for tasks with rewards r_i and r_j with max-ent action-value functions Q^i, Q^j. Define $C_b^\infty(s_t, a_t)$ as the fixed point of*

$$\begin{aligned} C_b^{(k+1)}(s_t,a_t) = -\alpha\gamma \mathbb{E}_{p(s_{t+1}|s_t,a_t)}\Big[\\ \log \int_{\mathcal{A}} \pi_i(a_{t+1}|s_{t+1})^b \pi_j(a_{t+1}|s_{t+1})^{(1-b)} \exp\big(\\ -\frac{1}{\alpha} C_b^{(k)}(s_{t+1},a_{t+1})\big) da_{t+1}\Big] \end{aligned}$$

Given the conditions for Soft Q convergence, the max-ent optimal $Q_b^*(s,a)$ for $r_b = br_i + (1-b)r_j$ is

$$\begin{aligned} Q_b^*(s,a) = bQ^i(s,a) + (1-b)Q^j(s,a) - C_b^\infty(s,a) \\ \forall s \in \mathcal{S}, a \in \mathcal{A}, b \in [0,1] \end{aligned}$$

We follow a similar approach to (Haarnoja et al., 2018a) but without making approximations and generalizing to all convex combinations.

First note that since π_i and π_j are optimal then $\pi_i(a|s) = \exp(\frac{1}{\alpha}(Q^i(s,a) - V^i(s)))$.

For brevity we use s and s' notation rather than writing the time index.

Define

$$Q_b^{(0)}(s,a) \equiv bQ^i(s,a) + (1-b)Q^j(s,a) \tag{17}$$

$$C^{(0)}(s,a) \equiv 0 \tag{18}$$

and consider soft Q-iteration on r_b starting from $Q_b^{(0)}$. We prove, inductively, that at each iteration $Q_b^{(k+1)} = bQ^i(s,a) + (1-b)Q^j(s,a) - C^{(k+1)}(s,a)$.

This is true by definition for $k = 0$.

$$Q_b^{(k+1)}(s,a) = r_b(s,a) + \gamma\alpha\mathbb{E}_{p(s'|s,a)}\left[\log\int_{\mathcal{A}}\exp\frac{1}{\alpha}Q_b^{(k)}(s',a')da'\right] \tag{19}$$

$$= r_b(s,a)+ \tag{20}$$

$$\gamma\alpha\mathbb{E}_{p(s'|s,a)}\left[\log\int_{\mathcal{A}}\exp(\frac{1}{\alpha}(bQ^i(s',a') + (1-b)Q^j(s',a') - C^{(k)}(s',a')))da'\right]$$

$$= r_b(s,a)+ \tag{21}$$

$$\mathbb{E}_{p(s'|s,a)}\left[bV^i(s') + (1-b)V^j(s') + \alpha\log\int_{\mathcal{A}}\exp(b\log\pi_i(a'|s') + (1-b)\log\pi_j(a'|s') - \frac{1}{\alpha}C^{(k)}(s',a'))da'\right]$$

$$= bQ^i(s,a) + (1-b)Q^j(s,a)+ \tag{22}$$

$$\alpha\gamma\mathbb{E}_{p(s'|s,a)}\left[\log\int_{\mathcal{A}}\exp(b\log\pi_1(a'|s') + (1-b)\log\pi_2(a'|s') - \frac{1}{\alpha}C^{(k)}(s',a'))da'\right]$$

$$= bQ^i(s,a) + (1-b)Q^i(s,a) - C_b^{(k+1)}(s,a). \tag{23}$$

Since soft Q-iteration converges to the α max-ent optimal soft Q then equation 17 holds at the limit.

One can get an intuition for $C_b^\infty(s,a)$ by noting that

$$C_b^{(1)}(s,a) = \gamma\alpha\mathbb{E}_{p(s'|s,a)}\left[(1-b)\,\mathrm{D}_b\left(\pi_1(\cdot|s)\|\pi_2(\cdot|s)\right)\right] \tag{24}$$

where D_b is the Rényi divergence of order b. $C_b^\infty(s,a)$ can be seen as the discount sum of divergences, weighted by the unnormalized product distribution $\pi_1(a|s)^b\pi_2(a|s)^{1-b}$.

A.3. N policies

It is possible to extend Theorem 3.2 to the case with N policies in a straightforward way.

Theorem A.1 (Multi-policy DC Optimality) *Let $\pi_1, \pi_2, ..., \pi_N$ be α max-ent optimal policies for tasks with rewards $r_1, r_2, ..., r_N$ with max-ent action-value functions $Q^1, Q^2, ..., Q^N$.*

Define $C_{\mathbf{w}}^\infty(s_t,a_t)$ as the fixed point of

$$C_{\mathbf{w}}^{(k+1)}(s_t,a_t) = -\alpha\gamma\mathbb{E}_{p(s_{t+1}|s_t,a_t)}\left[\log\int_{\mathcal{A}}\left(\prod_{i=1}^N\pi_i(a_{t+1}|s_{t+1})^{w_i}\right)\exp(-\tfrac{1}{\alpha}C_{\mathbf{w}}^{(k)}(s_{t+1},a_{t+1}))da_{t+1}\right]$$

Given the conditions for Soft Q convergence, the max-ent optimal $Q_{\mathbf{w}}^*(s,a)$ for the convex combination of rewards $r_{\mathbf{w}} = \sum_{i=1}^N r_i w_i$ is

$$Q_{\mathbf{w}}^*(s,a) = \textstyle\sum_{i=1}^N w_i Q^i(s,a) - C_{\mathbf{w}}^\infty(s,a)$$

$$\forall s\in\mathcal{S}, a\in\mathcal{A}, \mathbf{w}\in\{\mathbf{w}|\sum_{i=1}^N w_i = 1 \;\text{ and }\; w_i\geq 0\}$$

Note that w_i refers to component i of the vector $\mathbf{w}_i$.

The proof is very similar to the two reward case above.

Define

$$Q_{\mathbf{w}}^{(0)} \equiv \sum_{i=1}^N w_i Q^i(s,a) \tag{25}$$

$$C_{\mathbf{w}}^{(0)} \equiv 0 \tag{26}$$

and again consider soft Q-iteration on $r_{\mathbf{w}}$. We prove by induction that at each iteration

$$Q_{\mathbf{w}}^{(k+1)}(s,a) = \sum_{i=1}^{N} w_i Q^i(s,a) - C_{\mathbf{w}}^{(k+1)}(s,a) \tag{27}$$

Again, this is true by definition for $k = 0$. Now we consider a step of Soft Q iteration

$$\begin{aligned}
Q_{\mathbf{w}}^{(k+1)} &= r_{\mathbf{w}}(s,a) + \gamma\alpha\mathbb{E}_{p(s'|s,a)}\left[\log\int_{\mathcal{A}}\exp\frac{1}{\alpha}Q_{\mathbf{w}}^{(k)}(s',a')da'\right] && (28)\\
&= r_{\mathbf{w}}(s,a) + \gamma\alpha\mathbb{E}_{p(s'|s,a)}\left[\log\int_{\mathcal{A}}\exp\frac{1}{\alpha}\left(\sum_{i=1}^{N} w_i Q^i(s',a') - C_{\mathbf{w}}^{(k)}(s,a)\right)da'\right] && (29)\\
&= r_{\mathbf{w}}(s,a) + \gamma\mathbb{E}_{p(s'|s,a)}\left[\sum_{i=1}^{N} w_i V^i(s') + \alpha\log\int_{\mathcal{A}}\exp\left(\sum_{i=1}^{N} w_i\log\pi_i(a'|s') - \frac{1}{\alpha}C_{\mathbf{w}}^{(k)}(s',a')\right)da'\right] && (30)\\
&= \sum_{i=1}^{N} w_i Q^i(s,a) + \alpha\gamma\mathbb{E}_{p(s'|s,a)}\left[\log\int_{\mathcal{A}}\exp(\sum_{i=1}^{N} w_i\log\pi_i(a'|s') - \frac{1}{\alpha}C_{\mathbf{w}}^{(k)}(s',a'))da'\right] && (31)\\
&= \sum_{i=1}^{N} w_i Q^i(s,a) - C_{\mathbf{w}}^{(k+1)}(s,a) && (32)
\end{aligned}$$

Since soft Q-iteration converges to the α max-ent optimal soft Q then $Q_{\mathbf{w}}^{*}(s,a) = \sum_{i=1}^{N} w_i Q^i(s,a) - C_{\mathbf{w}}^{(k+1)}(s,a)$ for all $s \in \mathcal{S},\ a \in \mathcal{A}$.

Note that, in practice, estimating $C_{\mathbf{w}}^{\infty}$ may be more challenging for larger N. For compositions of many policies, GPI may be more practical.

B. Theoretical properties of the composition methods

Method	**Optimal**	**Bounded loss**	**Requires $\boldsymbol{\phi}$**	**Requires $f(s,a\|b)$**
CO				
CondQ	✓	na	✓	✓
GPI		✓	✓	
DC	✓	na		✓

Table 1. Theoretical properties of different approaches to max-ent transfer. The methods compared are: CO, CondQ, max-ent GPI (over a fixed, finite set of policies), and DC. The columns indicate whether the transfer policy is optimal, the regret of the transfer policy is bounded, whether rewards for all tasks $\boldsymbol{\phi}$ need to be observed simultaneously during training and whether the method requires learning a function conditional on the transfer task b, $f(s,a|b)$. DC is the only method that both recovers (in principle) the optimal policy and does not require observing ϕ during training.

C. Algorithm details

C.1. Transfer algorithm

Algorithm 2 AISBP transfer algorithm

Load trained parameters θ_Q, θ_q, $\theta_{\boldsymbol{\psi}}$, θ_C, θ_{Q_b}.
Accept transfer task parameter b, transfer $method \in$ CO, GPI, DC, CondQ.
while testing **do**
  Importance sample transfer policy $\pi_b(a|s) \propto \exp\frac{1}{\alpha}Q^{method}(s,a)$ with mixture proposal $p_b(a|s)_{\theta_q}$.
end while

C.2. All losses and estimators

We use neural networks to parametrize all quantities. For each policy we learn an action-value $Q_{\theta_Q}(s,a)$, value $V_{\theta_V}(s)$ and proposal distribution $q_{\theta_q}(a|s)$. We use target networks for the proposal distribution $q_{\theta'_q}(a|s)$ and value $V_{\theta'_V}(s)$.

Here we enumerate all of the losses and their estimators. We use temporal difference (TD(0)) learning for all the RL losses, so all losses are valid off-policy. We use a replay buffer R and learn by sampling minibatches of SARS tuples of size B, we index over the batch dimension with l and use s'_l to denote the state following s_l, so the tuple consists of (s_l, a_l, r_l, s'_l). For importance sampled estimators we sample N actions for each state s_l and use a_{lk} to denote sample k for state l.

We learn a set of n policies, one for each task in $\mathcal{T}$ indexed by i. However, we write the losses for a single policy and drop i for notational simplicity.

C.2.1. Proposal loss

The proposal loss minimizes the KL divergence between the Boltzmann distribution $\pi(a|s) \propto \exp(\frac{1}{2}Q(s,a))$ and the proposal distribution.

$$\mathcal{L}(\theta_q) = \mathbb{E}_R\left[\mathbb{E}_{a\sim\pi(\cdot|s)}[\log\pi(a|s_t) - \log q_{\theta_q}(a|s_t)]\right] \tag{33}$$

As described in the text, this loss is estimated using importance sampling with a mixture distribution $p(a|s)$ containing equally weighted components consisting of the target proposal distribution $q_{\theta'_q}(a|s)$ for all policies and the uniform distribution.

$$p(a|s) = \frac{1}{n+1}\left(\frac{1}{V^{\mathcal{A}}} + \sum_{i=1}^{n} q^i_{\theta'_q}(a|s)\right) \tag{34}$$

where $V^{\mathcal{A}}$ is the volume of the action space (which is always bounded in our case).

The proposal loss is estimated using self-normalized importance sampling

$$\mathcal{L}(\theta_q) \approx -\frac{1}{B}\sum_{k=1}^{B}\sum_{l=1}^{N} w_{kl}\log q_{\theta_q}(a|s_t), \tag{35}$$

$$w'_{kl} = \frac{\frac{1}{\alpha}(Q_{\theta_Q}(s_k,a_{kl}))}{p(a_{kl}|s_k)};\;\; w_{kl} = \frac{w_{kl'}}{\sum_{m=1}^{N} w'_{km}}. \tag{36}$$

C.2.2. Value loss

The soft value loss is

$$\mathcal{L}(\theta_V) = \mathbb{E}_R\left[\frac{1}{2}(V_{\theta_V}(s_t) - \alpha\log\int_{\mathcal{A}}\exp(\frac{1}{\alpha}Q_{\theta_Q}(s_t,a))da)^2\right] \tag{37}$$

We estimate this using importance sampling with the proposal distribution $q_{\theta_q}(a|s)$ which is trying to fit the policy π.

$$\mathcal{L}(\theta_V) \approx \frac{1}{2B}\sum_{l=1}^{B}\left(V_{\theta_V}(s_l) - \alpha\log Z\right)^2 \tag{38}$$

$$Z = \left[\frac{1}{N}\sum_{k=1}^{N}\frac{\exp(\frac{1}{\alpha}Q_{\theta_Q}(s_l,a_{lk}))}{q_{\theta_q}(a_{lk}|s_l)}\right] \tag{39}$$

C.2.3. ACTION-VALUE LOSS

The TD(0) loss for Q_{θ_Q} is

$$\mathcal{L}(\theta_Q) = \mathbb{E}_R\left[\frac{1}{2}(Q_{\theta_Q}(s_t,a_t) - (r(s_t,a_t,s_{t+1}) + \gamma V_{\theta'_V}(s_{t+1})))^2\right] \tag{40}$$

This does not require importance sampling to estimate and can be straightforwardly estimated as

$$\mathcal{L}(\theta_Q) \approx \frac{1}{2B}\sum_{l=1}^{B}(Q_{\theta_Q}(s_l,a_l) - (r_l + \gamma V_{\theta'_V}(s')))^2 \tag{41}$$

The action-value is parametrized as an advantage function $Q_{\theta_Q}(s,a) = V_{\theta'_V}(s) + A_{\theta_A}(s,a)$.

C.2.4. STATE DEPENDENT SUCCESSOR FEATURES LOSS

To facilitate max-ent GPI we learn successor features for each policy, both state-action dependent features $\boldsymbol{\psi}_{\theta_{\boldsymbol{\psi}}}(s,a)$ and state-dependent $\boldsymbol{\Upsilon}_{\theta_{\boldsymbol{\Upsilon}}}(s)$. As with value, we use a target network for the state-dependent features $\boldsymbol{\Upsilon}_{\theta'_{\boldsymbol{\Upsilon}}}(s)$

$$\mathcal{L}(\theta_{\boldsymbol{\Upsilon}}) = \mathbb{E}_R\left[\frac{1}{2}(\boldsymbol{\Upsilon}_{\theta_{\boldsymbol{\Upsilon}}}(s_t) - \mathbb{E}_{a_t\sim\pi(a_t|s_t)}[\boldsymbol{\psi}_{\theta_{\boldsymbol{\psi}}}(s_t,a_t) + \alpha\mathbf{1}(-Q_{\theta_Q}(s_t,a_t) + \alpha\log Z(s_t))])^2\right]$$

This loss is estimated using self-normalized importance sampling with proposal q_{θ_q}

$$\mathcal{L}(\theta_{\boldsymbol{\Upsilon}}) \approx \frac{1}{2B}\sum_{l=1}^{B}\sum_{k=1}^{N} w_{lk}\left[(\boldsymbol{\psi}^i_{\theta_{\boldsymbol{\psi}}}(s_l,a_{lk}) - Q^i_{\theta_Q}(s_l,a_{lk}) + \alpha\log Z(s_l))^2\right], \tag{42}$$

$$w_{lk} \propto \frac{\exp(\frac{1}{\alpha}Q^i(s_l,a_lk))}{q^i_{\theta_q}(a_{lk}|s_l)}. \tag{43}$$

We use the importance sampled estimate of Z from eq 39, rather than the value network which may be lagging the true partition function. We use self-normalized importance sampling to avoid the importance weights depending on $\alpha\log Z(s_l)$ (this introduces a bias, but in practise appears to work well).

C.2.5. STATE-ACTION DEPENDENT SUCCESSOR FEATURES LOSS

The state-action dependent successor feature loss is

$$\mathcal{L}(\theta_{\boldsymbol{\psi}}) = \mathbb{E}_R\left[\frac{1}{2}(\boldsymbol{\psi}_{\theta_Q}(s_t,a_t) - (\boldsymbol{\phi}(s_t,a_t,s_{t+1}) + \gamma\boldsymbol{\Upsilon}_{\theta'_{\boldsymbol{\Upsilon}}}(s_{t+1})))^2\right]. \tag{44}$$

for which we use the following estimator

$$\mathcal{L}(\theta_{\boldsymbol{\psi}^i}) \approx \frac{1}{2B}\sum_{l=1}^{B}(\boldsymbol{\psi}^i_{\theta_{\boldsymbol{\psi}}}(s_l,a_l) - (\boldsymbol{\phi}_l + \gamma\boldsymbol{\Upsilon}_{\theta'_{\boldsymbol{\Upsilon}}}(s'_l)))^2. \tag{45}$$

$\boldsymbol{\psi}_{\theta_{\boldsymbol{\psi}}}$ is parametrized as a "psi-vantage" $\boldsymbol{\psi}_{\theta_{\boldsymbol{\psi}}}(s,a) = \boldsymbol{\Upsilon}_{\theta'_{\boldsymbol{\Upsilon}}}(s) + \boldsymbol{\psi}^A_{\theta_A}(s,a)$.

C.2.6. DC CORRECTION

We learn the divergence correction for each pair of policies $\pi_i(a|s)$, $\pi_j(a|s)$. As described in the text, in order to learn $C_{\theta_C}(s,a,b)$ for all $b\in[0,1]$, we sample b. We also use a target network $C_{\theta'_C}(s,a,b)$. The loss is then

$$\begin{aligned}\mathcal{L}(\theta_C) = \mathbb{E}_{s\sim R, b\sim U(0,1)}[\tfrac{1}{2}(C_{\theta_C}(s,a,b) + \alpha\gamma\mathbb{E}_{p(s'|s,a)}[\log\textstyle\int_{\mathcal{A}}\exp(b\log\pi_i(a'|s') + \\ (1-b)\pi_j(a'|s') - \frac{1}{\alpha}C_{\theta_{C'}}(s',a',b))da'])^2].\end{aligned} \tag{46}$$

This loss is challenging to estimate, due to the dependence on two policies. We importance sample using a mixture of all proposal distributions uniform $p(a|s)$ (equation 34). We denote the samples of $b \sim \mathcal{U}(0,1)$ for each batch entry b_l. Note the choice of uniform distribution for b is not required, other distributions that ensure the estimator works well for $b \in [0,1]$ would also work. The importance sampled estimator is then

$$\mathcal{L}(\theta_C) \approx \frac{1}{N}\sum_{l=1}^{B}\left(C_{\theta_C}(s_l,a_l,b_l) - \alpha\gamma \log\left[\frac{1}{N}\sum_{k=1}^{N}\frac{C^{target}_{\theta'_C}(s'_l,a'_{lk},b_l)}{p(a_{lk'}|s_l)}\right]\right)^2, \tag{47}$$

$$C^{target}_{\theta'_C}(s'_l,a'_{lk},b_l) \equiv \exp(\frac{1}{\alpha}(b_l Q^i_{\theta_Q}(s'_l,a'_{lk}) + (1-b_l)Q^j_{\theta_Q}(s'_l,a'_{lk}) - C_{\theta'_C}(s'_l,a'_{lk},b_l)). \tag{48}$$

We parametrized C_{θ_C} as an advantage function $C_{\theta_C}(s,a,b) = C^A_{\theta_{CA}}(s,a,b) + C^B_{\theta_{CB}}(s,b)$ with an additional loss to constrain this parametrization

$$\mathcal{L}(\theta_B) = \mathbb{E}_{a\sim q(\cdot|s),s\sim R}\left[\frac{1}{2}(C^A_{\theta_{CA}}(s,a,b))^2\right] \tag{49}$$

which can be straightforwardly estimated by sampling from q

$$\mathcal{L}(\theta_B) \approx \frac{1}{2NB}\sum_{l=1}^{B}\sum_{k=1}^{N}(C^A_{\theta_{CA}}(s_l,a_{lk},b_l))^2 \tag{50}$$

C.2.7. CondQ

We also consider, as a control, learning the action-value function conditional on b directly (Schaul et al., 2015), in a similar way to the DC correction. We learn both a conditional value $V_{\theta_{V_b}}(s,b)$ and $Q_{\theta_{Q_b}}(s,a,b)$, again by sampling b uniformly each update.

$$\mathcal{L}(\theta_{V_b}) = \mathbb{E}_{R,b\sim U(0,1)}\left[\frac{1}{2}(V_{\theta_{V_b}}(s,b) - \alpha\log\int\exp(\frac{1}{\alpha}Q_{\theta_{Q_b}}(s,a,b)))^2\right], \tag{51}$$

$$\mathcal{L}\theta_Q = \mathbb{E}_{R,b\sim U(0,1)}\left[\frac{1}{2}(Q_{\theta_{Q_b}}(s,a,b) - (r_b + \gamma V_{\theta_{V_b}}(s',b)))^2\right], \tag{52}$$

where computing r_b for arbitrary b requires ϕ to have been observed.

We estimate Cond-Q with the same importance samples as C from $p(a|s)$ and again sample $b \sim \mathcal{U}(0,1)$ for each entry in the batch. We use target networks for $V_{\theta'_V}(s,b)$ and parametrize $Q_{\theta_Q}(s,a,b) = V_{\theta'_V}(s,b) + A_{\theta_A}(s,a,b)$.

The conditional value estimator is

$$\mathcal{L}(\theta_V) \approx \frac{1}{2B}\sum_{l=1}^{B}\left(V_{\theta_{V_b}}(s_l,b_l) - \alpha\log\frac{1}{N}\sum_{k=1}^{N}\frac{\exp(\frac{1}{\alpha}Q_{\theta_{Q_b}}(s_l,a_{lk},b_l)}{p(a_{lk}|s_l)}\right)^2 \tag{53}$$

and action-value estimator is

$$\mathcal{L}(\theta_Q) \approx \frac{1}{2B}\sum_{l=1}^{B}\left(Q_{\theta_{Q_b}}(s_l,a_l,b_l) - (r_b + \gamma V_{\theta'_{V_b}}(s'_l,b_l))\right)^2 \tag{54}$$

C.3. Sampling the product of proposals

The proposal distributions $q^i(a|s)$ are mixtures of M (truncated) normals (equation 7). We ignore the truncation when computing the product of proposals $q^{ij}(a|s)$.

The product of two M component mixtures of normals results in another mixture of normals with M^2 components (e.g. Schrempf et al., 2005). Since for all experiments M is a relatively small integer (maximum is 16) we sample from the product of proposals in a naive way.

D. Justification for the DC-Cheap heuristic

We wish to estimate $C_b^\infty(s,a)$ (defined in Theorem 3.2) while avoiding learning a conditional function of b. We make two (substantial) assumptions to arrive at this approximation.

Firstly, we assume policies $\pi_i(a|s), \pi_j(a|s)$ are Gaussian

$$\pi_i(a|s) = \exp\left(-\frac{(a-\mu_i(s))^2}{2\sigma(s)^2}\right) \tag{55}$$

and the variance $\sigma(s)$ is the same for both policies given a state (it may vary across states).

Secondly, we assume $C_b^{(k)}(s,a) = C_b^{(k)}(s)$ is independent of action. This is approximately correct when nearby states have similar Rényi divergences between policies.

We make use of a result by Gil et al. (2013) that states that the Rényi divergence of order b for two Gaussians of the same variance is

$$\mathrm{D}_b\left(\mathcal{N}(\mu_1,\sigma)\|\mathcal{N}(\mu_2,\sigma)\right) = \frac{1}{2}\frac{b(\mu_1-\mu_2)^2}{\sigma^2}. \tag{56}$$

We first define

$$G_b(s) \equiv (1-b)\,\mathrm{D}_b\left(\pi_i(\cdot|s)\|\pi_j(\cdot|s)\right) = -\log\int \pi_i(a|s)^b\pi_j(a|s)^{(1-b)}da. \tag{57}$$

From equation 55

$$G_b(s) = 4b(1-b)G_{\frac{1}{2}}(s). \tag{58}$$

Given these assumptions we show inductively that $C_b^{(k)}(s,a) = 4b(1-b)C_{1/2}^{(k)}(s,a)\ \forall k, b \in [0,1]$.

Since $C_b^{(0)}(s,a) = 0\ \forall b \in [0,1], a \in \mathcal{A}, s \in \mathcal{S}$ this is true for $k=0$. We show it holds inductively

$$C_b^{(k+1)}(s,a) = -\alpha\gamma\mathbb{E}_{p(s'|s,a)}\left[\log\int_{\mathcal{A}}\pi_i(a'|s')^b\pi_j(a'|s')^{(1-b)}\exp(-\frac{1}{\alpha}C_b^{(k)}(s',a'))da'\right] \tag{59}$$

$$= \gamma\mathbb{E}_{p(s'|s,a)}\left[\alpha G_b(s') + C_b^{(k)}(s')\right] \tag{60}$$

$$= 4b(1-b)C_{\frac{1}{2}}^{(k+1)}(s,a). \tag{61}$$

Obviously these assumptions are not justified. However, note that we estimate the true divergence for $C_{1/2}^\infty$, i.e. without any assumptions of Gaussian policies and this heuristic is used to estimate C_b^∞ from $C_{1/2}^\infty$. In practise, we find this heuristic works in many situations where the policies have similar variance, particulary when bounded by GPI.

E. Additional Figures

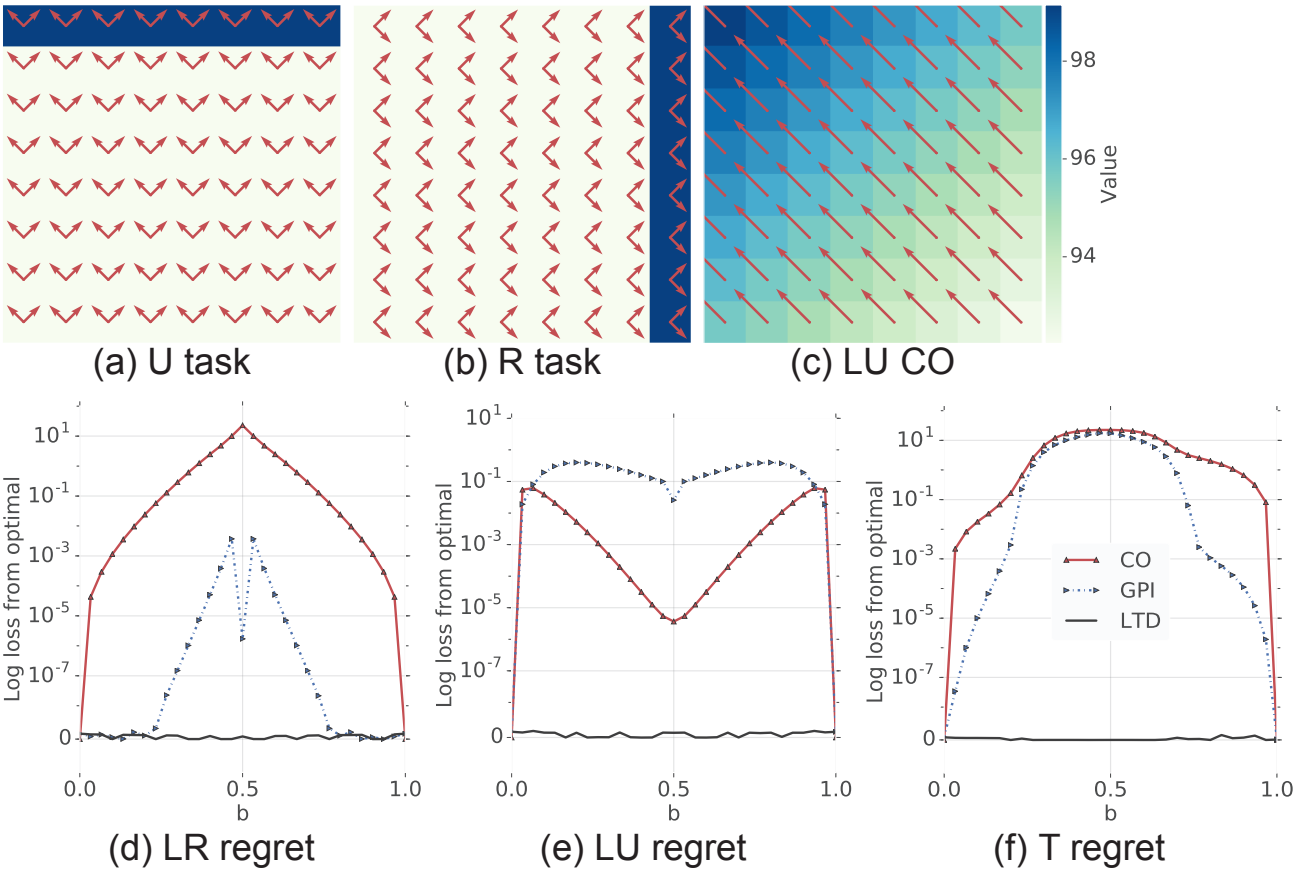

Figure 5. **Additional results for figure 1 (tabular)**
(**a**) The U(p) and (**b**) R(ight) tasks.
(**c**) The CO policy for the LU task. Note how even far from the reward (e.g. bottom right corner) the CO policy is near optimal, contrast with the GPI policy for this task (figure 1f).
The log regret (smaller is better) as function of b ($r_b = br_1 + (1-b)r_2$) for the transfer task for the (**d**) incompatible (Left-Right) task, (**e**) compatible (Left Up) task and (**f**) T(ricky) task.

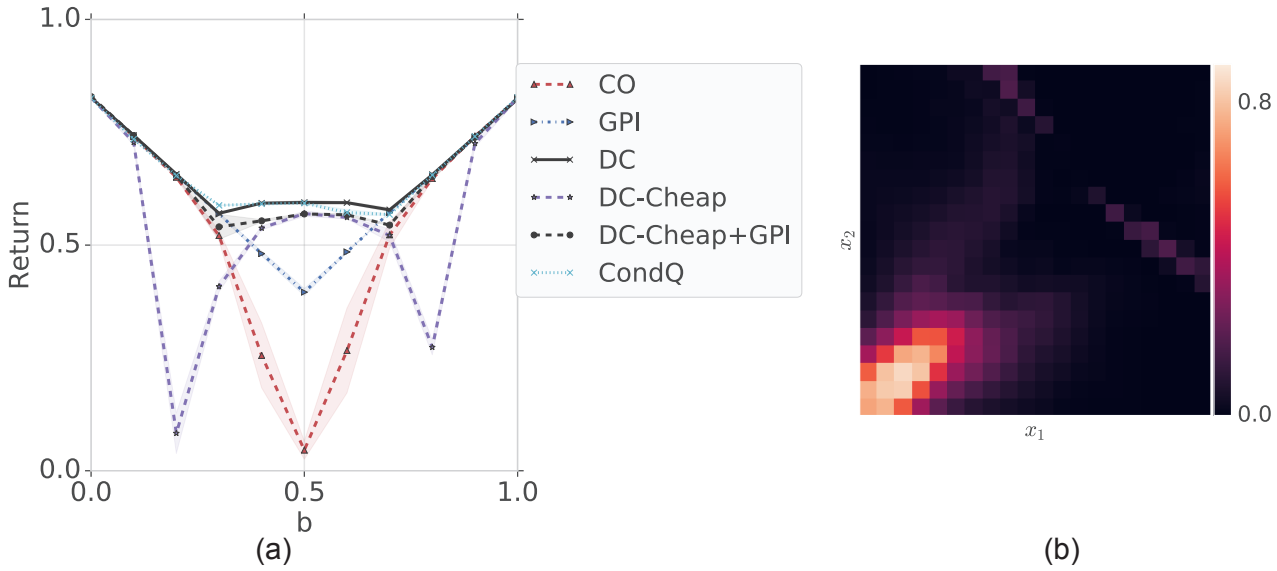

Figure 6. **Additional results for figure 2 (point mass tricky)**
(**a**) The returns (larger is better) for the transfer task as a function of b ($r_b = br_1 + (1-b)r_2$) including the DC heuristics. DC-Cheap+GPI performs almost as well as DC.
(**b**) The Rényi divergence of the two base policies as a function of position: the two policies are compatible except near the bottom left corner where the rewards are non-overlapping.

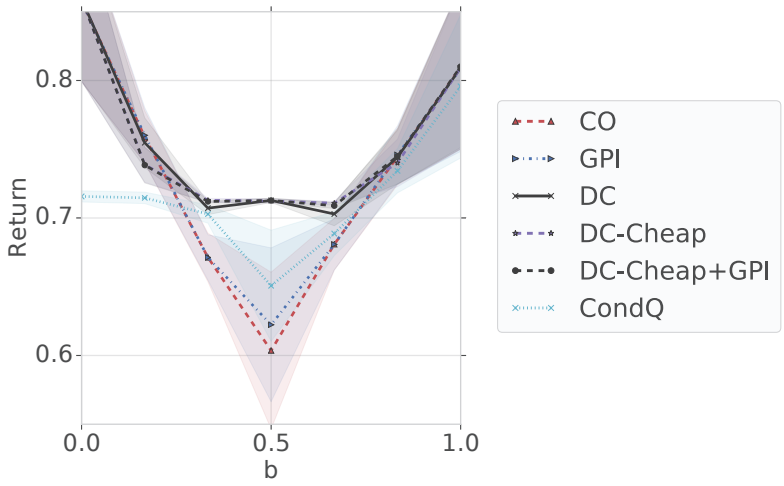

Figure 7. **Returns for figure 3 (planar manipulator)**
The returns for the transfer task as a function of b ($r_b = br_1 + (1-b)r_2$) including the DC heuristics. DC-Cheap+GPI performs almost as well as DC. Shaded bars show SEM (5 seeds).

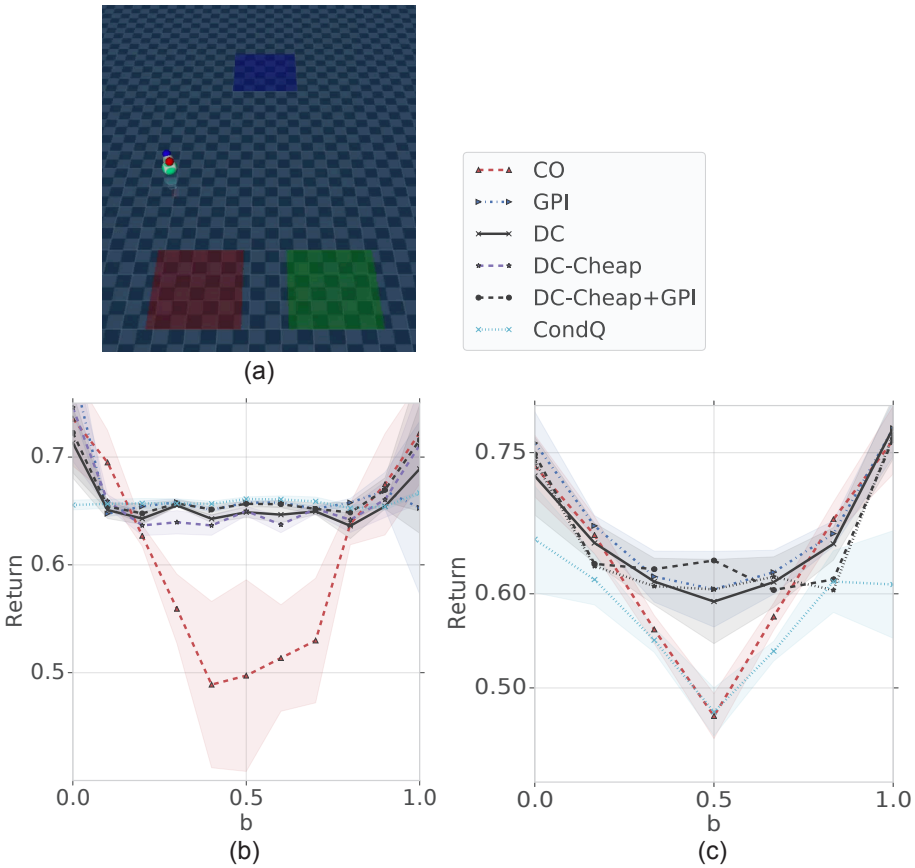

Figure 8. **Additional results for figure 4 (mobile bodies)**
(**a**) Jumping ball task. The task has rewards $(1, 0), (0, 1)$ in the green and red boxes respectively and $(0.75, 0.75)$ in the blue square. The returns for the transfer task as a function of b ($r_b = br_1 + (1 - b)r_2$) including the DC heuristics for the jumping ball (**b**) and ant (**c**). Shaded bars show SEM (5 seeds for ant, 3 seeds for jumping ball). As expected, CO performs poorly on these tasks. CondQ struggles to consistently get good returns on the ant task. The DC heuristics perform well on these tasks.

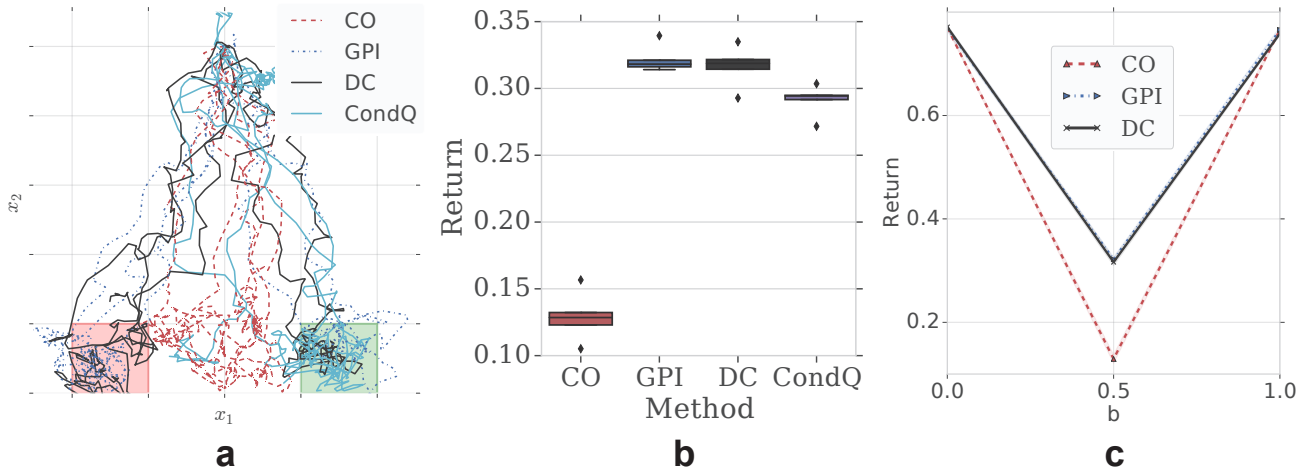

Figure 9. **Ant on non-composable subtasks**
(**a**) Trajectories of the ant during transfer on non-composable subtasks. In this experiment the two base tasks consists of rewards at the red and green square respectively. As expected, in this task, where the two base tasks have no compositional solution, CO (red) performs poorly with trajectories that end up between the two solutions. GPI (blue) performs well, as does DC (black). CondQ does slightly worse.
(**b**) Box-plot of returns from 5 seeds (at $b = 0.5$).
(**c**) Returns as a function of b, SEM across 5 seeds is plotted, but is smaller than the line thickness.

F. Experiment details

All control tasks were simulated using the MuJoCo physics simulator and constructed using the DM control suite (Tassa et al., 2018) which uses the MuJoCo physics simulator (Todorov et al., 2012).

The point mass was velocity controlled, all other tasks were torque controlled. The planar manipulator task was based off the planar manipulator in the DM control suite. The reward in all tasks was sparse as described in the main text.

During training for all tasks we start states from the randomly sampled positions and orientations. For the point mass, jumping ball and ant we evaluated transfer starting from the center (in the walker environments, the starting orientation was randomly sampled during transfer, the point mass does not have an orientation). For the planar manipulator transfer was tested from same random distribution as in training. Infinite time horizon policies were used for all tasks.

Transfer is made challenging by the need for good exploration. That was not the focus on this work. We aided exploration in several ways: during training we acted according to a higher-temperature policy $\alpha_e = 2\alpha$. We also sampled actions uniformly in an ϵ-greedy fashion with $\epsilon = 0.1$ and added Gaussian exploration noise during training. This was sufficient to explore the state space for most tasks. For the planar manipulator and the jumping ball, we found it necessary to induce behavior tasks by learning tasks for reaching the blue target. This behavior policy was, of course, only used for experience and not during transfer.

Below we list the hyper-parameters and networks use for all experiment. The discount γ and α were the only sensitive parameters that we needed to vary between tasks to adjust for the differing magnitudes of returns and sensitivity of the action space between bodies. If α is too small then the policies often only find one solution and all transfer approaches behave similarly, while for large α the resulting policies are too stochastic and do not perform well.

The state vector was preprocessed by a linear projection of $3\times$ its dimension and then a $\tanh$ non-linearity. All action-state networks (Q, ψ, C) consisted of 3 hidden layers with elu non-linearities (Clevert et al., 2015), with both action and preprocessed state projected by linear layers to be of the same dimensionality and used for input the first layer. All value networks and proposal networks consisted of 2 layers with elu non-linearities. The number of neurons in each layer was varied between environments, but was kept the same in all networks and layers (we did not sweep over this parameter, but choose a reasonable number based on our prior on the complexity of the task).

Below we list the per task hyper-parameters

Proposal learning rate	10^{-3}
All other learning rates	10^{-4}
Value target update period	200
Proposal target update period	200
Υ target update period	500
Number of importance samples for all estimators during learning	200
Number of importance samples for acting during training	50
Number of importance samples for acting during transfer	1000

Table 2. Parameters the same across all experiments

Task	**Number of units**	α	γ
Point mass	22	1	0.99
Planar Manipulator	192	0.05	0.99
Jumping Ball	192	0.2	0.9
Ant	252	0.1	0.95

Table 3. Parameters varied between experiments