Appendix

A. Notations

For better understanding, this section provides notation conversions for some of the key concepts in the main text beyond matrix-vector notation.

- Q-value

$$\mathbf{q}_\theta = \sum_{i=0}^{\infty} (\gamma P \Pi_\theta)^i \mathbf{r}$$

$$q_\theta(s,a) = \mathbb{E}_{A_i \sim \pi_\theta(S_i,\cdot), S_{i+1} \sim P(\cdot|S_i,A_i)} \left[\sum_{i=0}^{\infty} \gamma^i r(S_i, A_i) \middle| S_0 = s, A_0 = a \right]$$

- The stationary distribution's recursive form (3)

$$\mathbf{d}_\theta = (1-\gamma)\Pi_\theta^\top \boldsymbol{\mu}_0 + \gamma \Pi_\theta^\top P^\top \mathbf{d}_\theta$$

$$d_\theta(s,a) = (1-\gamma)\mu_0(s)\pi_\theta(s,a) + \gamma \sum_{\tilde{s},\tilde{a}} d_\theta(\tilde{s},\tilde{a}) P(s|\tilde{s},\tilde{a}) \pi_\theta(s,a)$$

- The cumulative reward objective (6), actor objective (10) and critic objective (11):

$$\begin{aligned}
J(\theta) &= (1-\gamma)\boldsymbol{\mu}_0^\top \Pi_\theta \sum_{i=0}^{\infty} (\gamma P \Pi_\theta)^i \mathbf{r} && = (1-\gamma)\mathbb{E}_{S_0 \sim \mu_0, A_i \sim \pi_\theta(S_i,\cdot), S_{i+1} \sim P(\cdot|S_i,A_i)} \left[\sum_{i=0}^{\infty} \gamma^i r(S_i, A_i) \right] \\
&= (1-\gamma)\boldsymbol{\mu}_0^\top \Pi_\theta \mathbf{q}_\theta && = (1-\gamma)\mathbb{E}_{S_0 \sim \mu_0, A_0 \sim \pi_\theta(S_0,\cdot)} [q_\theta(S_0, A_0)] \\
&= \mathbf{d}_\theta^\top \mathbf{r} && = \mathbb{E}_{(S,A) \sim d_\theta} [r(S,A)] \\
J_\pi(\boldsymbol{\theta}, \boldsymbol{\phi}) &= (1-\gamma)\boldsymbol{\mu}_0^\top \Pi_\theta \mathbf{q}_\phi && = (1-\gamma)\mathbb{E}_{S_0 \sim \mu_0, A_0 \sim \pi_\theta(S_0,\cdot)} [q_\phi(S_0, A_0)] \\
J_q(\boldsymbol{\theta}, \boldsymbol{\phi}) &= \frac{1}{2} \|\mathbf{r} + \gamma P \Pi_\theta \mathbf{q}_\phi - \mathbf{q}_\phi\|_{\mathbf{d}}^2 && = \frac{1}{2} \mathbb{E}_{(S,A) \sim d} \left[\left(r(S,A) + \gamma \mathbb{E}_{S' \sim P(\cdot|S,A), A' \sim \pi_\theta(S',\cdot)} [q_\phi(S', A')] - q_\phi(S,A) \right)^2 \right]
\end{aligned}$$

- The policy gradient (8, 25), actor$_g$ (9, 25) and actor$_o$ (12) gradients

$$\begin{aligned}
\nabla_\theta J &= H_\theta \Delta(\Xi^\top \mathbf{d}_{\mathcal{S},\theta}) \mathbf{q}_\theta && = \sum_s d_{\mathcal{S},\theta}(s) \sum_a q_\theta(s,a) \partial_\theta \pi_\theta(s,a) \\
\nabla_\theta^\phi J &= H_\theta \Delta(\Xi^\top \mathbf{d}_{\mathcal{S},\theta}) \mathbf{q}_\phi && = \sum_s d_{\mathcal{S},\theta}(s) \sum_a q_\phi(s,a) \partial_\theta \pi_\theta(s,a) \\
\partial_\theta J_\pi &= (1-\gamma) H_\theta \Delta(\Xi^\top \boldsymbol{\mu}_0) \mathbf{q}_\phi && = (1-\gamma) \sum_s \mu_0(s) \sum_a q_\phi(s,a) \partial_\theta \pi_\theta(s,a)
\end{aligned}$$

- The gap between policy gradient and actor gradients (21, 29)

$$\begin{aligned}
\nabla_\theta J &= \nabla_\theta^\phi J + \partial_\theta (\mathbf{d}_\theta^\top \boldsymbol{\delta}'_{\theta,\phi}) && = \nabla_\theta^\phi J + \partial_\theta \mathbb{E}_{(S,A) \sim d_\theta} [\delta'_{\theta,\phi}(S,A)] \\
&= \partial_\theta J_\pi + \partial_\theta (\mathbf{d}_\theta^\top \boldsymbol{\delta}_{\theta,\phi}) && = \partial_\theta J_\pi + \partial_\theta \mathbb{E}_{(S,A) \sim d_\theta} [\delta_{\theta,\phi}(S,A)]
\end{aligned}$$

B. Proofs

Theorem 2. *[Stationary distribution derivative] Let the derivative matrix Υ of the stationary distribution w.r.t. the policy parameters to be $\Upsilon_{i,sa} = \frac{\partial d_\theta(s,a)}{\partial \theta_i}$. Then*

$$\Upsilon = H_\theta \Delta(\Xi^\top \mathbf{d}_{\mathcal{S},\theta}) \Psi_\theta^{-1}.$$

Proof. By the chain rule, we can calculate Υ as

$$\Upsilon = H_\theta \widetilde{\Upsilon} \tag{49}$$

where $\widetilde{\Upsilon}_{\tilde{s}\tilde{a},sa} = \frac{\partial d_\theta(s,a)}{\partial \pi_\theta(\tilde{s},\tilde{a})}$ and recall that $(H_\theta)_{i,sa} = \frac{\partial \pi_\theta(s,a)}{\partial \theta_i}$ as defined in Eq.(2). Next we show how to calculate $\widetilde{\Upsilon}$ using the implicit function theorem.

Based on Eq.(3), define $\mathbf{f}(\mathbf{d}, \boldsymbol{\pi})$ as

$$\mathbf{f}(\mathbf{d}, \boldsymbol{\pi}) = (I - \gamma \Pi^\top P^\top)\mathbf{d} - (1-\gamma)\Pi^\top \boldsymbol{\mu}_0 \tag{50}$$

We know from Eq.(3) that $\mathbf{f}(\mathbf{d}_\theta, \boldsymbol{\pi}_\theta) = \mathbf{0}_{|\mathcal{S}||\mathcal{A}|}$. The Jacobian w.r.t. $\mathbf{d}$ at $(\mathbf{d}_\theta, \boldsymbol{\pi}_\theta)$ is

$$\left.\frac{\partial \mathbf{f}}{\partial \mathbf{d}}\right|_{\mathbf{d}=\mathbf{d}_\theta, \boldsymbol{\pi}=\boldsymbol{\pi}_\theta} = I - \gamma \Pi_\theta^\top P^\top = \Psi_\theta^\top \tag{51}$$

which is invertible under regular assumptions (recall that $\gamma < 1$). As for $\frac{\partial \mathbf{f}}{\partial \boldsymbol{\pi}}\big|_{\mathbf{d}=\mathbf{d}_\theta, \boldsymbol{\pi}=\boldsymbol{\pi}_\theta}$, we can see that it is diagonal because

$$f(s,a) = d(s,a) - \gamma \pi(s,a) \sum_{\tilde{s},\tilde{a}} P(s|\tilde{s},\tilde{a}) d(\tilde{s},\tilde{a}) - (1-\gamma)\pi(s,a)\mu_0(s) \tag{52}$$

and it does not depend on policy values other than $\pi(s,a)$. Then

$$\left.\frac{\partial f(s,a)}{\partial \pi(s,a)}\right|_{\mathbf{d}=\mathbf{d}_\theta, \boldsymbol{\pi}=\boldsymbol{\pi}_\theta} = -\gamma \sum_{\tilde{s},\tilde{a}} P(s|\tilde{s},\tilde{a}) d_\theta(\tilde{s},\tilde{a}) - (1-\gamma)\mu_0(s) = -d_{\mathcal{S},\theta}(s) \tag{53}$$

where the last equality is due to Eq.(3). Broadcasting it to all actions, we get

$$\left.\frac{\partial \mathbf{f}}{\partial \boldsymbol{\pi}}\right|_{\mathbf{d}=\mathbf{d}_\theta, \boldsymbol{\pi}=\boldsymbol{\pi}_\theta} = -\Delta(\Xi^\top \mathbf{d}_{\mathcal{S},\theta}) \tag{54}$$

Then by the implicit function theorem, we have

$$\widetilde{\Upsilon}^\top = -\left(\left.\frac{\partial \mathbf{f}}{\partial \mathbf{d}}\right|_{\mathbf{d}=\mathbf{d}_\theta, \boldsymbol{\pi}=\boldsymbol{\pi}_\theta}\right)^{-1} \left(\left.\frac{\partial \mathbf{f}}{\partial \boldsymbol{\pi}}\right|_{\mathbf{d}=\mathbf{d}_\theta, \boldsymbol{\pi}=\boldsymbol{\pi}_\theta}\right) \tag{55}$$

$$= (\Psi_\theta^\top)^{-1} \Delta(\Xi^\top \mathbf{d}_{\mathcal{S},\theta}) \tag{56}$$

$$\implies \widetilde{\Upsilon} = \Delta(\Xi^\top \mathbf{d}_{\mathcal{S},\theta}) \Psi_\theta^{-1} \tag{57}$$

This combined with Eq.(49) completes the proof. □

Theorem 3. *Under Assumption 1,*

$$\mathbf{g}_{S,\theta} = \partial_\theta J_\pi + \partial_\theta(\mathbf{d}_\theta^\top \boldsymbol{\delta}_{\theta,\phi}) = \nabla_\theta J.$$

Proof. The first term can be computed as

$$\partial_q J_\pi = (1-\gamma)\Pi_\theta^\top \boldsymbol{\mu}_0. \tag{58}$$

Plugging Eqs.(3), (39) and (58) to Eq.(38) gives

$$\mathbf{g}_{S,\theta} = \partial_\theta J_\pi - (\partial_\theta \partial_q J_q)^\top (\Psi_\theta^\top D_\theta \Psi_\theta)^{-1} (I - \gamma \Pi_\theta^\top P^\top)\mathbf{d}_\theta \tag{59}$$

$$= \partial_\theta J_\pi - (\partial_\theta \partial_q J_q)^\top \Psi_\theta^{-1} D_\theta^{-1} \mathbf{d}_\theta \tag{60}$$

$$= \partial_\theta J_\pi - \frac{1}{1-\gamma} (\partial_\theta \partial_q J_q)^\top \mathbf{1}_{|\mathcal{S}||\mathcal{A}|} \tag{61}$$

where the last equation is due to

$$\Psi_\theta^{-1}\mathbf{1}_{|\mathcal{S}||\mathcal{A}|} = \left(\sum_{i=0}^{\infty}(\gamma P\Pi_\theta)^i\right)\mathbf{1}_{|\mathcal{S}||\mathcal{A}|} = \sum_{i=0}^{\infty}\gamma^i\left((P\Pi_\theta)^i\mathbf{1}_{|\mathcal{S}||\mathcal{A}|}\right) = \sum_{i=0}^{\infty}\gamma^i\mathbf{1}_{|\mathcal{S}||\mathcal{A}|} = \frac{1}{1-\gamma}\mathbf{1}_{|\mathcal{S}||\mathcal{A}|} \tag{62}$$

Under natural regularity assumptions that allow transposing derivative with summation, we can compute the gradient using Eq.(13) as

$$\begin{aligned}
\mathbf{g}_{S,\theta} &= \partial_\theta J_\pi + \frac{1}{1-\gamma}\left[\partial_\theta\left(\Psi_\theta^\top D_\theta\boldsymbol{\delta}_\theta\right)\right]^\top\mathbf{1}_{|\mathcal{S}||\mathcal{A}|} && (63)\\
&= \partial_\theta J_\pi + \frac{1}{1-\gamma}\partial_\theta\left[\left(\Psi_\theta^\top D_\theta\boldsymbol{\delta}_\theta\right)^\top\mathbf{1}_{|\mathcal{S}||\mathcal{A}|}\right] && (64)\\
&= \partial_\theta J_\pi + \partial_\theta\left[\boldsymbol{\delta}_\theta^\top D_\theta\mathbf{1}_{|\mathcal{S}||\mathcal{A}|}\right] \qquad \text{(Note that } \Psi_\theta\mathbf{1}_{|\mathcal{S}||\mathcal{A}|} = (1-\gamma)\mathbf{1}_{|\mathcal{S}||\mathcal{A}|}\text{)} && (65)\\
&= \partial_\theta J_\pi + \partial_\theta(\mathbf{d}_\theta^\top\boldsymbol{\delta}_\theta) && (66)
\end{aligned}$$

Therefore, the Stackelberg gradient $\mathbf{g}_{S,\theta}$ is not only doing (partial) policy improvement, but also maximizing $\mathbf{d}_\theta^\top\boldsymbol{\delta}_\theta$.

Adding this term to the actor objective, apply Eq.(3), and get the Stackelberg actor objective

$$\begin{aligned}
\max_\theta\ J_{S,\pi}(\theta,\mathbf{q}) &= (1-\gamma)\boldsymbol{\mu}_0^\top\Pi_\theta\mathbf{q} + \mathbf{d}_\theta^\top\boldsymbol{\delta}_\theta && (67)\\
&= \mathbf{d}_\theta^\top\Psi_\theta\mathbf{q} + \mathbf{d}_\theta^\top(\mathbf{r} - \Psi_\theta\mathbf{q}) && (68)\\
&= \mathbf{d}_\theta^\top\mathbf{r} && (69)
\end{aligned}$$

which is exactly the dual objective of the cumulative reward objective in Eq.(6). This means the Stackelberg gradient is unbiased policy gradient. □

C. Soft Actor-Critic

C.1. Derivation of Res-SAC

In this section, we derive the actor update in Res-SAC.

With an additional entropy term, the cumulative reward objective of SAC is given by

$$\begin{aligned}
J^{\mathrm{SAC}}(\theta) &= (1-\gamma)\boldsymbol{\mu}_0^\top\Pi_\theta\sum_{i=0}^{\infty}(\gamma P\Pi_\theta)^i(\mathbf{r} - \log\boldsymbol{\pi}_\theta) && (70)\\
&= (1-\gamma)\boldsymbol{\mu}_0^\top\Pi_\theta\left[\left(\mathbf{r} + \sum_{i=1}^{\infty}(\gamma P\Pi_\theta)^i(\mathbf{r} - \log\boldsymbol{\pi}_\theta)\right) - \log\boldsymbol{\pi}_\theta\right] && (71)\\
&= (1-\gamma)\boldsymbol{\mu}_0^\top\Pi_\theta(\mathbf{q}_\theta - \log\boldsymbol{\pi}_\theta) && (72)\\
&= \mathbf{d}_\theta^\top(\mathbf{r} - \log\boldsymbol{\pi}_\theta) && (73)
\end{aligned}$$

where $\mathbf{q}_\theta := \mathbf{r} + \sum_{i=1}^{\infty}(\gamma P\Pi_\theta)^i(\mathbf{r} - \log\boldsymbol{\pi}_\theta)$ is the action value accounting for all future entropy terms but *excluding* the current entropy term (as defined as in the SAC paper). Using a critic $\mathbf{q}_\phi$, the actor and critic objectives are

$$\begin{aligned}
\text{Actor}:\quad &\max_\theta\ J_\pi^{\mathrm{SAC}}(\theta,\phi) = (1-\gamma)\boldsymbol{\mu}_0^\top\Pi_\theta(\mathbf{q}_\phi - \log\boldsymbol{\pi}_\theta) && (74)\\
\text{Critic}:\quad &\min_\phi\ J_q^{\mathrm{SAC}}(\theta,\phi) = \frac{1}{2}\|\mathbf{r} + \gamma P\Pi_\theta(\mathbf{q}_\phi - \log\boldsymbol{\pi}_\theta) - \mathbf{q}_\phi\|_{\mathbf{d}}^2 && (75)
\end{aligned}$$

Note that the original SAC paper uses a KL divergence minimization step for the actor update (Haarnoja et al., 2018a, Eq.(10)), which can be seen as a variant of Eq.(74). More specifically, one can re-express the KL divergence as a Bregman divergence associated with the negative entropy $-\mathcal{H}$ (Nachum et al., 2017):

$$\max_{\theta} -\mathbb{E}_{S\sim\mu_0}\left[\mathrm{KL}\left(\boldsymbol{\pi}_\theta(S,\cdot)\middle\|\frac{\exp(\mathbf{q}_\phi(S,\cdot))}{Z_\phi(S)}\right)\right] = \mathbb{E}_{S\sim\mu_0}\left[-\boldsymbol{\pi}_\theta(S,\cdot)^\top\nabla\mathcal{H}\left(\frac{\exp(\mathbf{q}_\phi(S,\cdot))}{Z_\phi(S)}\right)+\mathcal{H}(\boldsymbol{\pi}_\theta(S,\cdot))+C\right] \tag{76}$$

$$= \mathbb{E}_{S\sim\mu_0}\left[\boldsymbol{\pi}_\theta(S,\cdot)^\top\mathbf{q}_\phi(S,\cdot)+\mathcal{H}(\boldsymbol{\pi}_\theta(S,\cdot))+C'\right] \tag{77}$$

$$= \boldsymbol{\mu}_0^\top\Pi_\theta(\mathbf{q}_\phi-\log\boldsymbol{\pi}_\theta)+C' \tag{78}$$

where $Z_\phi(s)$ is the partition function for state s and C, C' are some constants independent of θ. Thus Eq.(74) is the same as the KL divergence minimization up to some constants and rescaling.

The first order derivatives of Eqs.(74) and (75) are

$$\partial_\theta J_\pi^{\mathrm{SAC}} = (1-\gamma)\mathbb{E}_{S\sim\mu_0}\left[\partial_\theta\sum_A \pi_\theta(S,A)(q_\phi(S,A)-\log\pi_\theta(S,A))\right] \tag{79}$$

$$= (1-\gamma)\mathbb{E}_{S\sim\mu_0,\epsilon}\left[\partial_\theta\log\pi_\theta(S,A)+\partial_A(q_\phi(S,A)-\log\pi_\theta(S,A))\partial_\theta\pi_\theta(S,\epsilon)\right] \tag{80}$$

$$\partial_\phi J_\pi^{\mathrm{SAC}} = (1-\gamma)\Pi_\theta^\top\boldsymbol{\mu}_0 \tag{81}$$

$$\partial_\phi J_q^{\mathrm{SAC}} = -\Psi_\theta^\top D\boldsymbol{\delta}_E \tag{82}$$

where Eq.(80) uses the reparametrization trick (Schulman et al., 2015a) with $a=\pi_\theta(s,\epsilon)$ for some random variable ϵ, and $\boldsymbol{\delta}_E=\mathbf{r}+\gamma P\Pi_\theta(\mathbf{q}_\phi-\log\boldsymbol{\pi}_\theta)-\mathbf{q}_\phi$ is the residual of the critic accounting for the entropy of the next state. The reparametrization trick is needed because the policy is predicting an action in a continuous action space.

The gap between $J^{\mathrm{SAC}}(\theta)$ and $J_\pi^{\mathrm{SAC}}(\theta,\boldsymbol{\phi})$ is

$$J^{\mathrm{SAC}}(\theta)-J_\pi^{\mathrm{SAC}}(\theta,\boldsymbol{\phi}) = \mathbf{d}_\theta^\top(\mathbf{r}-\log\boldsymbol{\pi}_\theta)-(1-\gamma)\boldsymbol{\mu}_0^\top\Pi_\theta(\mathbf{q}_\phi-\log\boldsymbol{\pi}_\theta) \tag{83}$$

$$= \mathbf{d}_\theta^\top(\mathbf{r}-\log\boldsymbol{\pi}_\theta)-\mathbf{d}_\theta^\top\Psi_\theta(\mathbf{q}_\phi-\log\boldsymbol{\pi}_\theta) \tag{84}$$

$$= \mathbf{d}_\theta^\top\boldsymbol{\delta}_E \tag{85}$$

The question now becomes how to compute $\partial_\theta(\mathbf{d}_\theta^\top\boldsymbol{\delta}_E)$. As in Section 4.1 (20), by the product rule, we have

$$\partial_\theta(\mathbf{d}_\theta^\top\boldsymbol{\delta}_E) = \partial_\theta((\mathbf{d}_\theta')^\top\boldsymbol{\delta}_E)+\partial_\theta(\mathbf{d}_\theta^\top\boldsymbol{\delta}_E') \tag{86}$$

Using the reparametrization trick, $\partial_\theta((\mathbf{d}_\theta')^\top\boldsymbol{\delta}_E)$ can be computed as

$$\mathbb{E}_{(S,A)\sim d_\theta}[\partial_\theta\delta_E(S,A)] = \mathbb{E}_{(S,A)\sim d_\theta,\tilde{S}\sim P}\left[\gamma\partial_\theta\sum_{\tilde{A}}\pi_\theta(\tilde{S},\tilde{A})(q_\phi(\tilde{S},\tilde{A})-\pi_\theta(\tilde{S},\tilde{A}))\right] \tag{87}$$

$$= \gamma\mathbb{E}_{(S,A)\sim d_\theta,\tilde{S}\sim P,\tilde{\epsilon}}\left[\partial_\theta\log\pi_\theta(\tilde{S},\tilde{A})+\partial_{\tilde{A}}(q_\phi(\tilde{S},\tilde{A})-\log\pi_\theta(\tilde{S},\tilde{A}))\partial_\theta\pi_\theta(\tilde{S},\tilde{\epsilon})\right] \tag{88}$$

It can be combined with Eq.(80), using the Eq.(3), to get

$$\partial_\theta[(1-\gamma)\boldsymbol{\mu}_0^\top\Pi_\theta(\mathbf{q}_\phi-\log\boldsymbol{\pi}_\theta)+(\mathbf{d}_\theta')^\top\boldsymbol{\delta}_E] = \mathbb{E}_{(S,A)\sim d_\theta}\left[\partial_\theta\log\pi_\theta(S,A)+\partial_A(q_\phi(S,A)-\log\pi_\theta(S,A))\partial_\theta\pi_\theta(S,\epsilon)\right] \tag{89}$$

Eq.(89) explains the original SAC implementation for the actor update, except for using $\mathbf{d}_\theta$ instead of a replay buffer to compute the expectation. The final term, $\partial_\theta(\mathbf{d}_\theta^\top\boldsymbol{\delta}_E')$, is optimizing a policy to maximize $\boldsymbol{\delta}_E$ as a fixed reward. It is also the residual reward for learning a res-critic for Res-SAC.

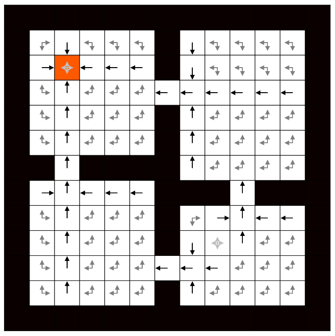
C.2. Derivation of Stack-SAC

In this section, we derive the actor update in Stack-SAC.

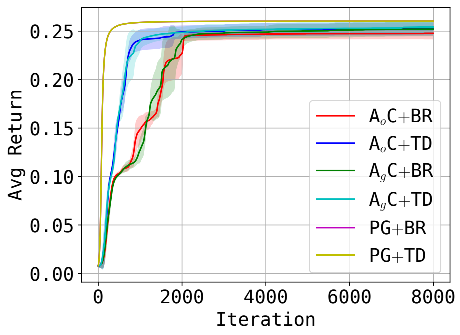
The second order derivative $\partial_\phi^2 J_q$ remains the same as Eq.(39). The Stackelberg gradient now becomes

$$\mathbf{g}_{S,\theta}^{\mathrm{SAC}} = \partial_\theta J_\pi+\partial_\theta(\Psi_\theta^\top D\boldsymbol{\delta}_E)^\top\Psi_\theta^{-1}D\mathbf{d}_\theta = \partial_\theta J_\pi+\partial_\theta(\mathbf{d}_\theta^\top\boldsymbol{\delta}_E). \tag{90}$$

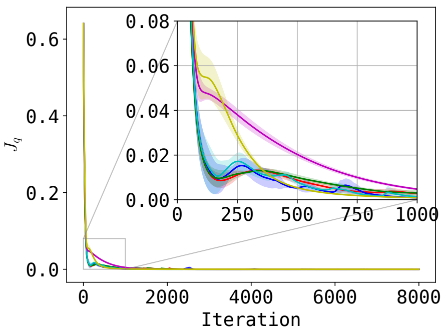
Then based on Eq.(85), $\mathbf{g}_{S,\theta}^{\mathrm{SAC}}=\nabla_\theta J^{\mathrm{SAC}}$ is an unbiased policy gradient for SAC.



(a) The FourRoom environment



(b) Policy performance J_π



(c) Objective value J_q

Figure 4: FourRoom with a goal and results of dynamic programming

D. Experiment Details

D.1. Tabular

Fig. 4a shows the FourRoom environment where goal is to reach a particular cell. The initial state distribution is a uniform distribution over all unoccupied cells. The reward is 1 for reaching the goal and 0 otherwise.

D.1.1. Additional Dynamic Programming Experiments

We investigate the effects of different gradient methods (Actor$_o$, Actor$_g$ or Policy Gradient (PG)) combined with different critic updates (Bellman residual minimization (BR) or temporal difference iteration (TD)) in the dynamic programming setting, where the reward $\mathbf{r}$ and the transition P are assumed to be known. This showcases the performance of different algorithms in the ideal scenario.

Specifically, the parameters of the softmax policy are the logits, and the critic is directly parametrized (i.e., it is tabular with a scaler value for each state-action pair). Using $\mathbf{r}$ and P, one can compute the Actor$_o$ gradient $\partial_\theta J_\pi$ (12), Actor$_g$ gradient $\nabla_\theta^\phi J$ (9) and PG (8) directly, and apply them to the policy parameters. As for the critic, BR uses the full gradient (13) while TD uses semi-gradient (18). The critic TD error loss is weighted according to the on-policy distribution $\mathbf{d}_\theta$. Both the actor and the critic use Adam optimizer, with respective learning rates of 0.01 and 0.02.

Figs. 4b and 4c show the average return of the policy J_π and the critic objective value J_q respectively. There are a few observations. (1) Policy gradient quickly converges to the optimal performance, regardless of whether BR or TD is used. The performance of PG+BR and PG+TD are hardly distinguishable in Fig. 4b, showing that if one can estimate PG accurately, the critic update may be less important. (2) Actor-Critic (AC) is slower than PG to achieve optimal performance. Even with much more iterations, both Actor$_o$-Critic (A$_o$C) and Actor$_o$-Critic (A$_g$C) are very slow to reach the final performance of PG. Furthermore, this happens even when the critic is providing accurate estimates of the Q-values (as J_q is very small after 2000 iterations). This indicates that PG can be a better choice as long as one has access to it, and our effort of closing the gap between AC and PG is meaningful in practice. (3) TD performs better than BR for both A$_o$C and A$_g$C. However, this is not always the case (Scherrer, 2010).

D.1.2. Sample-Based Experiments

This section refers to the FourRoom experiments in Section 6.1 in the main text. We implement sample-based algorithms which follow the Actor$_o$-Critic (A$_o$C), Actor$_g$-Critic (A$_g$C), Stack-AC, and Res-AC updates. Hyper-parameters are listed in Table 2.

A$_o$C uses the following procedure. We use two replay buffers, a replay buffer $\mathcal{O}$ which will store states from the initial state distribution of the environment, and a replay buffer $\mathcal{D}$ which will store transitions from the most recent episode run using the current policy. Concretely, $\mathcal{O}$ stores samples from $\boldsymbol{\mu}_0$ whereas $\mathcal{D}$ stores samples from the on-policy distribution $\mathbf{d}_\theta$. At the beginning of the training procedure, we initialize $\mathcal{O}$ to be empty. At the beginning of each episode, we initialize $\mathcal{D}$ to be empty, and we add the initial state sampled from the environment to the initial state replay buffer $\mathcal{O}$. For each step in the environment before the episode terminates, we add the current transition to the replay buffer $\mathcal{D}$. After the episode

Hyperparameters for four-room domain experiments	
Parameter	Value
optimizer	Adam
learning rate for actor/policy	$1 \cdot 10^{-2}$
learning rate for critic	$2 \cdot 10^{-2}$
discount (γ)	0.9
environment steps per gradient update	300 (episode length)
batch size	300
Stack-AC: value of η	0.5
Res-AC: learning rate for res-critic	$2 \cdot 10^{-2}$

Table 2: Hyperparameters used for Actor$_o$-Critic, Actor$_g$-Critic, Stack-AC, and Res-AC for the sample-based experiments on the four-room domain.

Algorithm 1 Actor$_o$-Critic

```
Initialize $\theta$, $\phi$
$\mathcal{O} \leftarrow \varnothing$
for each episode do
    $\mathcal{D} \leftarrow \varnothing$
    $s_0 \sim \mu_0$
    $\mathcal{O} \leftarrow \mathcal{O} \cup \{s_0\}$
    for each environment step do
        $a_t \sim \pi(a_t|s_t)$
        $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$
        $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1}, a_{t+1})\}$
    end for
    $\theta \leftarrow \theta + \alpha_\theta \partial_\theta \widetilde{J}^o_\pi$
    $\phi \leftarrow \phi - \alpha_\phi \partial_\phi \widetilde{J}_q$
end for
```

Algorithm 2 Actor$_g$-Critic

```
Initialize $\theta$, $\phi$
for each episode do
    $\mathcal{D} \leftarrow \varnothing$
    for each environment step do
        $a_t \sim \pi(a_t|s_t)$
        $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$
        $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1}, a_{t+1})\}$
    end for
    $\theta \leftarrow \theta + \alpha_\theta \partial_\theta \widetilde{J}^g_\pi$
    $\phi \leftarrow \phi - \alpha_\phi \partial_\phi \widetilde{J}_q$
end for
```

Algorithm 3 Stack-AC

```
Initialize $\theta$, $\phi$
$\mathcal{O} \leftarrow \varnothing$
for each episode do
    $\mathcal{D} \leftarrow \varnothing$
    $s_0 \sim \mu_0$
    $\mathcal{O} \leftarrow \mathcal{O} \cup \{s_0\}$
    for each environment step do
        $a_t \sim \pi(a_t|s_t)$
        $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$
        $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1}, a_{t+1})\}$
    end for
    $\theta \leftarrow \theta + \alpha_\theta \widetilde{\mathbf{g}}^{\text{semi}}_{S,\theta}$
    $\phi \leftarrow \phi - \alpha_\phi \partial_\phi \widetilde{J}_q$
end for
```

Algorithm 4 Res-AC

```
Initialize $\theta$, $\phi$
for each episode do
    $\mathcal{D} \leftarrow \varnothing$
    for each environment step do
        $a_t \sim \pi(a_t|s_t)$
        $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$
        $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1}, a_{t+1})\}$
    end for
    $\theta \leftarrow \theta + \alpha_\theta \partial_\theta \widetilde{J}^{\text{res}}_\pi$
    $\phi \leftarrow \phi - \alpha_\phi \partial_\phi \widetilde{J}_q$
    $\psi \leftarrow \psi - \alpha_\psi \partial_\psi \widetilde{J}_w$
end for
```

terminates, we apply gradient updates to the actor (policy) and the critic. Stack-AC follows this same procedure but with different actor updates. A_gC and Res-AC follow the same procedure except for not having an initial state buffer and using different actor updates. Res-AC additionally includes a res-critic update. For all algorithms, we compute a single critic update (and res-critic update for Res-AC) for each actor-update to product Fig. 1. To produce Fig. 2, we updated the res-critic 5 times for each critic update to obtain a more accurate res-critic in order to illustrate the res-critic's ability to close the gap between the critic's prediction and the true return.

For all algorithms we compute the gradient update for the critic as follows. We sample a batch of transitions $\mathcal{B}_D$ from the replay buffer $\mathcal{D}$ and compute the following loss function for the critic to minimize:

$$\widetilde{J}_q = \frac{1}{|\mathcal{B}_D|} \sum_{(s,a,r,\tilde{s},\tilde{a}) \in \mathcal{B}_D} (q_\phi(s,a) - (r + \gamma q'_\phi(\tilde{s},\tilde{a})))^2$$

where $\tilde{s}$, $\tilde{a}$ are the next state and next action in the transition, and $q'_\phi(\tilde{s},\tilde{a})$ indicates that no gradients pass through $q_\phi(\tilde{s},\tilde{a})$, i.e. it is treated as a target network. The gradient of this loss, $\partial_\phi \widetilde{J}_q$, is a sample-based estimate of $\partial_\phi J_q$ (13). Note, however, that it uses a semi-gradient instead of Bellman residual minimization in J_q (which yields the full/total gradient).

Actor$_o$-Critic

To compute the gradient update for the actor in A_oC, we sample a batch of initial states $\mathcal{B}_O$ from the replay buffer $\mathcal{O}$ and compute the following objective function for the actor to maximize:

$$\widetilde{J}^o_\pi = \frac{1}{|\mathcal{B}_O|} \sum_{s \in \mathcal{B}_O} \log \pi_\theta(a|s) q_\phi(s,a).$$

where the action a is sampled from the current policy π_θ for each state s in $\mathcal{B}_O$. The gradient of this objective, $\partial_\theta \widetilde{J}^o_\pi$, is a sample-based estimate of $\partial_\theta J_\pi$ (12). For the complete pseudocode of A_oC, see Algorithm 1.

Actor$_g$-Critic

To compute the gradient update for the actor in A_gC, we sample a batch of transitions $\mathcal{B}_D$ from the replay buffer $\mathcal{D}$ and compute the following objective for the actor to maximize:

$$\widetilde{J}^g_\pi = \frac{1}{|\mathcal{B}_D|} \sum_{(s,a,r,\tilde{s},\tilde{a}) \in \mathcal{B}_D} \log \pi_\theta(a|s) q_\phi(s,a).$$

The gradient of this objective, $\partial_\theta \widetilde{J}^g_\pi$, is a sample-based estimate of $\nabla^\phi_\theta J$ (9). For the complete pseudocode of Actor$_g$-Critic, see Algorithm 2.

Stack-AC

We compute the gradient update for the actor as follows. Then the Stackelberg gradient based on semi-critic-gradient is given by

$$\mathbf{g}^{\text{semi}}_{S,\theta} := \partial_\theta J_\pi - (\partial_\theta \partial^{\text{semi}}_q J_q)^\top ((\partial^{\text{semi}}_q)^2 J_q)^{-1} (\partial^{\text{semi}}_q J_\pi)$$

In the above equation, we replace J_q with $\widetilde{J}_q$ and J_π with $\widetilde{J}_\pi$. This gives us our actor update for Stack-AC:

$$\widetilde{\mathbf{g}}^{\text{semi}}_{S,\theta} := \partial_\theta \widetilde{J}_\pi - (\partial_\theta \partial^{\text{semi}}_q \widetilde{J}_q)^\top ((\partial^{\text{semi}}_q)^2 \widetilde{J}_q)^{-1} (\partial^{\text{semi}}_q \widetilde{J}_\pi)$$

For the complete pseudocode of Stack-AC, see Algorithm 3.

Res-AC

We compute the gradient update for the res-critic as follows. We sample a batch of transitions $\mathcal{B}_D$ from the replay buffer $\mathcal{D}$ and compute the following loss function for the res-critic to minimize:

$$\widetilde{J}_w = \frac{1}{|\mathcal{B}_D|} \sum_{(s,a,r,\tilde{s},\tilde{a}) \in \mathcal{B}_D} (w_\psi(s,a) - (\delta'_\theta(s,a) + \gamma w'_\psi(\tilde{s},\tilde{a})))^2$$

where $\delta'_\theta(s,a) = r(s,a) + \gamma q'_\phi(\tilde{s},\tilde{a}) - q'_\phi(s,a)$ is the TD-error computed using the current critic. Note that $w'_\psi(\tilde{s},\tilde{a})$ indicates that that no gradients pass through $w_\psi(\tilde{s},\tilde{a})$, i.e. it is treated as a target network. The gradient of this loss, $\partial_\psi \tilde{J}_w$, is a sample-based estimate of the gradient of J_w (31). Note, however, that it uses a semi-gradient instead of Bellman residual minimization in J_w (which yields the full/total gradient).

To compute the gradient update for the actor in Res-AC, we sample a batch of transitions $\mathcal{B}_D$ from the replay buffer $\mathcal{D}$ and compute the following objective for the actor to maximize:

$$\tilde{J}^{\text{res}}_\pi = \frac{1}{|\mathcal{B}_D|} \sum_{(s,a,r,\tilde{s},\tilde{a}) \in \mathcal{B}_D} \log \pi_\theta(a|s) q_\phi(s,a) + \log \pi_\theta(a|s) w_\psi(s,a).$$

The gradient of this objective, $\partial_\theta \tilde{J}^{\text{res}}_\pi$, is a sample-based estimate of the actor update in Res-AC (33). For the complete pseudocode of Res-AC, see Algorithm 4.

D.2. Continuous Control

Algorithm 5 Res-SAC

```
1: Initialize parameters $\theta$, $\phi$, $\psi$
2: Initialize replay buffer $\mathcal{D} \leftarrow \varnothing$
3: for each iteration do
4:     for each environment step do
5:         $a_t \sim \pi_\theta(a_t|s_t)$
6:         $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$
7:         $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1}, a_{t+1})\}$
8:     end for
9:     for each critic step do
10:        $\phi \leftarrow \phi - \alpha_\phi \hat{\nabla} J_Q(\phi)$
11:    end for
12:    for each res-critic step do
13:        $\psi \leftarrow \psi - \alpha_\psi \hat{\nabla} J_W(\psi)$
14:    end for
15:    for each actor step do
16:        $\theta \leftarrow \theta - \alpha_\theta \hat{\nabla} J_\pi(\theta)$
17:    end for
18: end for
```

Our training protocol for SAC, Stack-SAC, and Res-SAC follows the same training protoocl of SAC ((Haarnoja et al., 2018b)). Hyperparameters used for all algorithms are listed in Table 3.

D.2.1. Res-SAC: Loss Functions

Below, we present the loss functions and updates for the actor, critic, and res-critic of Res-SAC.

Similar to SAC, Res-SAC uses a parametrized soft Q-function (critic) $Q_\phi(s,a)$, and a tractable policy (actor) $\pi_\theta(a|s)$. Additionally, Res-SAC uses a parametrized residual Q-function (res-critic) $W_\psi(s,a)$. The parameters of these networks are ϕ, θ, and ψ.

The soft Q-function parameters are trained exactly as in SAC (Haarnoja et al., 2018b), but we write the objectives again here for clarity. The soft Q-function (critic) parameters are trained to minimize the soft Bellman residual:

$$J_Q(\phi) = \mathbb{E}_{(S,A)\sim\mathcal{D}} \left[\frac{1}{2} \left(Q_\phi(S,A) - \left(r(S,A) + \gamma \mathbb{E}_{S'\sim P, A'\sim\pi_\theta(S')} \left[Q_{\bar{\phi}}(S',A') - \log \pi_\theta(A'|S') \right] \right) \right)^2 \right]$$

where $\mathcal{D}$ is a replay buffer containing previously sampled states and actions, and the target soft Q-function $Q_{\bar{\phi}}$ uses an exponential moving average $\bar{\phi}$ of ϕ as done in the original SAC.

The residual Q-function have a similar objective, but the key differences are (1) the reward is based on the TD error of the critic and (2) the there is no entropy term. Specifically, the residual Q-function (res-critic) parameters are trained to

Hyperparameters for continuous control experiments	
Parameter	Value
optimizer	Adam
learning rate	$3 \cdot 10^{-4}$
discount (γ)	0.99
replay buffer size	10^6
number of hidden layers	2
number of hidden units per layer	128
number of samples per minibatch	128
nonlinearity	ReLU
target smoothing coefficient (τ)	0.005
target update interval	1
entropy target	$-\dim(\mathcal{A})$
environment steps per gradient step	10
Stack-SAC: value of η	0.5
Res-SAC: value of c	6.0 for HalfCheetah-v2, 1.0 for Reacher-v2, 4.0 for Pendulum-v0

Table 3: Hyperparameters used for SAC, Stack-SAC, and Res-SAC for continuous control experiments.

minimize the Bellman residual:

$$J_W(\psi) = \mathbb{E}_{(S,A)\sim\mathcal{D}}\left[\frac{1}{2}\left(W_\psi(S,A) - \left(\tilde{r}(S,A) + \gamma\mathbb{E}_{S'\sim P, A'\sim\pi_\theta(S')}\left[W_{\bar{\psi}}(S',A')\right]\right)\right)^2\right]$$

where $\bar{\psi}$ is an exponential moving average of ψ. The clipped reward $\tilde{r}(s,a)$ is computed as follows:

$$\tilde{r}(s,a) = \text{clip}(\delta(s,a), -c, c) = \min(\max(\delta(s,a), -c), c) \qquad \text{for } c > 0$$

where

$$\delta(s,a) = r(s,a) + \gamma\mathbb{E}_{S'\sim P, A'\sim\pi_\theta(S')}[Q_{\bar{\phi}}(S',A')] - Q_\phi(s,a)$$

The actor / policy is trained by minimizing the KL divergence:

$$J_\pi(\theta) = \mathbb{E}_{S\sim\mathcal{D}}\left[\text{KL}\left(\pi_\theta(\cdot|S)\middle\|\frac{\exp\left[Q_\phi(S,\cdot) + Q_\psi(S,\cdot)\right]}{Z_{\phi,\psi}(S)}\right)\right]$$

All objectives above can be optimized with stochastic gradients: $\hat{\nabla}J_Q(\phi)$, $\hat{\nabla}J_W(\psi)$, and $\hat{\nabla}J_\pi(\theta)$. The pseudocode for Res-SAC can be found in Algorithm 5.

D.2.2. Res-SAC: Sensitivity Analysis

The results in Section 6.2 suggest that actor-critic algorithms enjoy improved sample efficiency and final performance when their actor update rules are modified to follow the Res-AC framework. In the continuous control tasks, we found that the performance of Res-SAC was dependent on the setting of an additional hyper-parameter: the clip value $c > 0$ applied to the TD error δ_θ that is used as the reward for the res-critic. On HalfCheetah-v2, we examine the sensitivity of Res-SAC to the clip value on the res-critic's reward (Fig.5). Without clipping, training is highly unstable. Higher clip values improve stability, with a clip value of 6.0 leading to the best performance. We found a useful heuristic to select a clip value for Res-SAC: train SAC on the same task and use the maximum absolute TD error of the critic that occurs during training as the clip value c for Res-SAC. As an example, we see that when training SAC on HalfCheetah-v2, the max TD error is between 5.0 and 6.0 (Fig.6), and we find that a clip value of 6.0 leads to the best performance of Res-SAC on HalfCheetah.

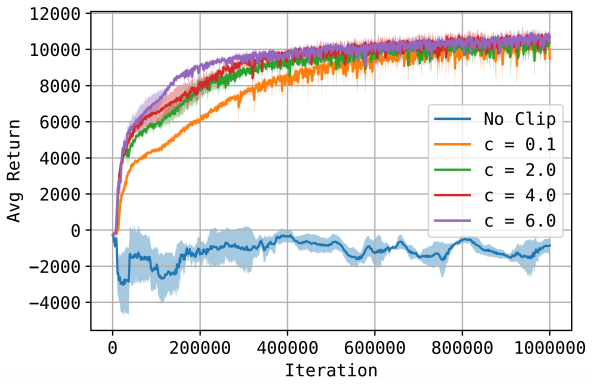


Figure 5: Sensitivity of Res-SAC to the clip value c on the HalfCheetah-v2 task.

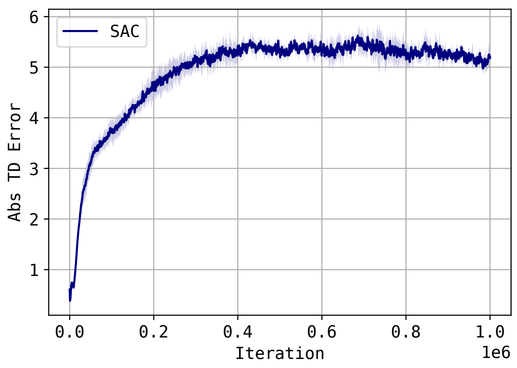


Figure 6: Absolute TD error while training SAC on HalfCheetah-v2.