A COLLABORATIVE Q DETAILS

We derive the gradient and provide the training details for Eq. 5.

Gradient for Training Objective. Taking derivative w.r.t θ_n^a and θ_n^c in Eq. 5, we arrive at the following gradient:

$$\nabla_{\theta_n^a}\mathbf{L}_n(\theta_n^a,\theta_n^c) = \mathbb{E}_{s_i,a\sim\rho(\cdot),r_i;s'\sim\varepsilon}[(r+\gamma\max_{a'}Q_i(s^{'},a^{'},r_i;\theta_{n-1}^a,\theta_{n-1}^c)-Q_i(o_i,a,r_i;\theta_n^a,\theta_n^c)) \nabla_{\theta_n^a}Q_i^a(s_i,a,r_i;\theta_n^a)] \tag{6a}$$

$$\nabla_{\theta_n^c}\mathbf{L}_n(\theta_n^a,\theta_n^c) = \mathbb{E}_{s_i,a\sim\rho(\cdot),r_i;s'\sim\varepsilon}[(r+\gamma\max_{a'}Q_i(s^{'},a^{'},r_i;\theta_{n-1}^a,\theta_{n-1}^c)-Q_i(o_i,a,r_i;\theta_n^a,\theta_n^c)) \nabla_{\theta_n^c}Q_i^c(o_i,a,r_i;\theta_n^c)-\alpha Q_i^c(s_i,a,r_i;\theta_n^c)\nabla_{\theta_n^c}Q_i^c(s_i,a,r_i;\theta_n^c)] \tag{6b}$$

Soft CollaQ. In the actual implementation, we use a soft-constraint version of CollaQ: we subtract $Q^{\text{collab}}(o_i^{alone},a_i)$ from Eq. 4. The *Q-value Decomposition* now becomes:

$$Q_i(o_i,a_i) = Q_i^{\text{alone}}(o_i^{alone},a_i)+Q_i^{\text{collab}}(o_i,a_i)-Q^{\text{collab}}(o_i^{alone},a_i) \tag{7}$$

The optimization objective is kept the same as in Eq. 5. This helps reduce variances in all the settings in resource collection and Starcraft multi-agent challenge. We sometimes also replace $Q^{\text{collab}}(o_i^{alone},a_i)$ in Eq. 7 by its target to further stabilize training.

B ENVIRONMENT SETUP AND TRAINING DETAILS

Resource Collection. We set the discount factor as 0.992 and use the RMSprop optimizer with a learning rate of 4e-5. ϵ-greedy is used for exploration with ϵ annealed linearly from 1.0 to 0.01 in $100k$ steps. We use a batch size of 128 and update the target every 10k steps. For temperature parameter α, we set it to 1. We run all the experiments for 3 times and plot the mean/std in all the figures.

StarCraft Multi-Agent Challenge. We set the discount factor as 0.99 and use the RMSprop optimizer with a learning rate of 5e-4. ϵ-greedy is used for exploration with ϵ annealed linearly from 1.0 to 0.05 in $50k$ steps. We use a batch size of 32 and update the target every 200 episodes. For temperature parameter α, we set it to 0.1 for 27m_vs_30m and to 1 for all other maps.

All experiments on StarCraft II use the default reward and observation settings of the SMAC benchmark. For ad hoc team play with different VIP, an additional 100 reward is added to the original 200 reward for winning the game if the VIP agent is alive after the episode.

For swapping agent types, we design the maps 3s1z_vs_16zg, 1s3z_vs_16zg and 2s2z_vs_16zg (**s** stands for stalker, **z** stands for zealot and **zg** stands for zergling). We use the first two maps for training and the third one for testing. For adding units, we use 27m_vs_30m for training and 28m_vs_30m for testing (**m** stands for marine). For removing units, we use 29m_vs_30m for training and 28m_vs_30m for testing.

We run all the experiments for 4 times and plot the mean/std in all the figures.

C DETAILED RESULTS FOR RESOURCE COLLECTION

We compare CollaQ with QMIX and CollaQ with attention-based model in resource collection setting. As shown in Fig. 9, QMIX doesn't show great performance as it is even worse than random action. Adding attention-based model introduces a larger variance, so the performance degrades by 10.66 in training but boosts by 2.13 in ad ad hoc team play.

D DETAILED RESULTS FOR STARCRAFT MULTI-AGENT CHALLENGE

We provide the win rates for CollaQ and QMIX on the environments without random agent IDs on three maps. Fig. 10 shows the results for both method.

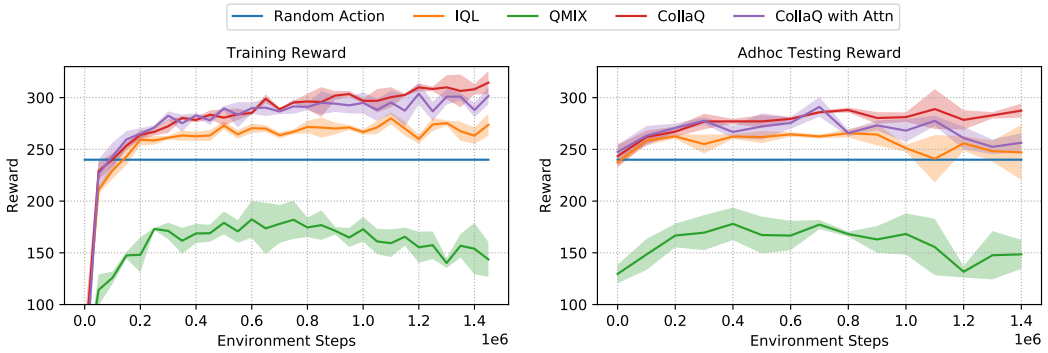

Figure 9: Results for resource collection. Adding attention-based model to CollaQ introduces a larger variance so the performance is a little worse. QMIX doesn't show good performance in this setting.

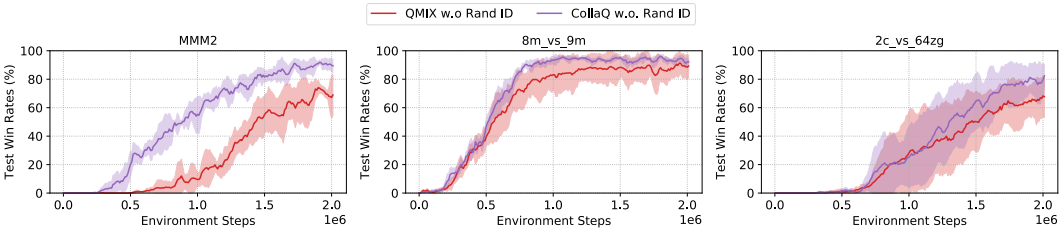

Figure 10: Results for StarCraft Multi-Agent Challenge without random agent IDs. CollaQ outperforms QMIX on all three maps.

We show the exact win rates for all the maps and settings mentioned in StarCraft Multi-Agent Challenge. From Tab. 1, we can clearly see that CollaQ improves the previous SoTA by a large margin.

Table 1: Win rates for StarCraft Multi-Agent Challenge. CollaQ show superior performance over all baselines.

	IQL	VDN	QTRAN	QMIX	CollaQ	CollaQ with Attn
5m_vs_6m	62.81%	69.37%	35.31%	66.25%	**81.88%**	80.00%
MMM2	4.22%	6.41%	0.32%	36.56%	79.69%	**84.69%**
2c_vs_64zg	33.75%	22.66%	8.13%	34.06%	**87.03%**	62.66%
27m_vs_30m	1.10%	6.88%	0.00%	19.06%	41.41%	**50.63%**
8m_vs_9m	71.09%	82.66%	28.75%	77.97%	92.19%	**96.41%**
10m_vs_11m	70.47%	86.56%	31.10%	81.10%	91.25%	**97.50%**

We also check the margin of winning scenarios, measured as how many units survive after winning the battle. The experiments are repeated over 128 random seeds. CollaQ surpasses the QMIX by over 2 units on average (Tab. 2), which is a huge gain.

Table 2: Number of survived units on six StaCraft maps. We compute mean and standard deviation over 128 runs. CollaQ outperforms all baselines significantly by managing more units to survive.

	5m_vs_6m	MMM2	2c_vs_64zg	27m_vs_30m	8m_vs_9m	10m_vs_11m
IQL	0.91 $\pm$ 0.28	0.02 $\pm$0.03	0.05 $\pm$ 0.04	0.00 $\pm$ 0.00	0.95 $\pm$ 0.36	0.6 $\pm$ 0.44
VDN	1.35 $\pm$ 0.13	0.28 $\pm$ 0.32	0.23 $\pm$ 0.12	0.55 $\pm$ 0.93	3.16 $\pm$ 0.61	3.39 $\pm$ 1.44
QTRAN	1.76 $\pm$ 0.53	0.31 $\pm$ 0.44	0.36 $\pm$ 0.35	0.00 $\pm$ 0.00	2.43 $\pm$ 0.53	3.06 $\pm$ 2.11
QMIX	1.72 $\pm$ 0.5	1.92 $\pm$ 1.02	0.47 $\pm$ 0.11	1.79 $\pm$ 0.72	2.75 $\pm$ 0.48	3.89 $\pm$ 1.74
CollaQ	1.95 $\pm$ 0.41	**4.89** $\pm$ 1.32	**1.48** $\pm$ 0.15	2.80 $\pm$ 0.94	**3.98** $\pm$ 0.56	**4.91** $\pm$ 1.48
CollaQ with Attn	**2.77** $\pm$ 0.17	4.73 $\pm$ 1.08	1.00 $\pm$ 0.49	**5.22** $\pm$ 1.79	3.68 $\pm$ 0.63	4.73 $\pm$ 0.41

In a simple ad hoc team play setting, we assign a new VIP agent whose survival matters at test time. Results in Tab. 3 show that at test time, the VIP agent in CollaQ has substantial higher survival rate than QMIX.

Table 3: VIP agents survival rates for StarCraft Multi-Agent Challenge. CollaQ with attention surpasses QMIX by a large margin.

	IQL	VDN	QTRAN	QMIX	CollaQ	CollaQ with Attn
5m_vs_6m	30.47%	46.72%	16.72%	38.13%	56.72%	**61.72**%
MMM2	0.31%	0.63%	0.16%	30.16%	62.34%	**81.41**%
8m_vs_9m	37.35%	47.34%	6.25%	48.91%	59.06%	**78.13**%

We also test CollaQ in a harder ad hoc team play setting: swapping/adding/removing agents at test time. Tab 4 summarizes the results for ad hoc team play, CollaQ outperforms QMIX by a lot.

Table 4: Win rates for StarCraft Multi-Agent Challenge with swapping/adding/removing agents. CollaQ improves QMIX substantially.

	IQL	VDN	QTRAN	QMIX	CollaQ	CollaQ with Attn
Swapping	0.00%	18.91%	0.00%	37.03%	46.25%	**46.41**%
Adding*	13.44%	23.28%	0.16%	70.94%	-	**79.22**%
Removing*	0.94%	16.41%	0.16%	58.44%	-	**73.12**%

* IQL, VDN, QTRAN and QMIX here all use attention-based models.

E VIDEOS AND VISUALIZATIONS OF STARCRAFT MULTI-AGENT CHALLENGE

We extract several video frames from the replays of CollaQ's agents for better visualization. In addition to that, we provide the full replays of QMIX and CollaQ. CollaQ's agents demonstrate super interesting behaviors such as healing the agents under attack, dragging back the unhealthy agents, and protecting the VIP agent (under the setting of ad hoc team play with different VIP agent settings). The visualizations and videos are available at `https://sites.google.com/view/collaq-starcraft`

F PROOF AND LEMMAS

Lemma 1. *If* $a_1' \geq a_1$, *then* $0 \leq \max(a_1', a_2) - \max(a_1, a_2) \leq a_1' - a_1$.

Proof. Note that $\max(a_1, a_2) = \frac{a_1+a_2}{2} + \left|\frac{a_1-a_2}{2}\right|$. So we have:

$$\max(a_1', a_2) - \max(a_1, a_2) = \frac{a_1' - a_1}{2} + \left|\frac{a_1' - a_2}{2}\right| - \left|\frac{a_1 - a_2}{2}\right| \leq \frac{a_1' - a_1}{2} + \left|\frac{a_1 - a_1'}{2}\right| = a_1' - a_1 \tag{8}$$

□

F.1 LEMMAS

Lemma 2. *For a Markov Decision Process with finite horizon* H *and discount factor* $\gamma < 1$. *For all* $i \in \{1, \ldots, K\}$, *all* $\mathbf{r}_1, \mathbf{r}_2 \in \mathbb{R}^M$, *all* $s_i \in S_i$, *we have:*

$$|V_i(s_i; \mathbf{r}_1) - V_i(s_i; \mathbf{r}_2)| \leq \sum_{x,a} \gamma^{|s_i - x|} |r_1(x, a) - r_2(x, a)| \tag{9}$$

where $|s_i - x|$ *is the number of steps needed to move from* s_i *to* x.

Proof. By definition of optimal value function V_i for agent i, we know it satisfies the following Bellman equation:

$$V_i(x_h; \mathbf{r}_i) = \max_{a_i} \left(r_i(x_i, a_i) + \gamma \mathbb{E}_{x_{h+1}|x_h, a_h} \left[V_i(x_{h+1})\right]\right) \tag{10}$$

Note that to avoid confusion between agents initial states $\mathbf{s} = \{s_1, \ldots, s_K\}$ and reward at state-action pair (s, a), we use (x, a) instead. For terminal node x_H, which exists due to finite-horizon MDP with horizon H, $V_i(x_H) = r_i(x_H)$. The current state s_i is at step 0 (i.e., $x_0 = s_i$).

We first consider the case that $\mathbf{r}_1$ and $\mathbf{r}_2$ only differ at a single state-action pair (x_h^0, a_h^0) for $h \leq H$. Without loss of generality, we set $r_1(x_h^0, a_h^0) > r_2(x_h^0, a_h^0)$.

By definition of finite horizon MDP, $V_i(x_{h'}; \mathbf{r}_1) = V_i(x_{h'}; \mathbf{r}_2)$ for $h' > h$. By the property of max function (Lemma 1), we have:

$$0 \leq V_i(x_h^0; \mathbf{r}_1) - V_i(x_h^0; \mathbf{r}_2) \leq r_1(x_h^0, a_h^0) - r_2(x_h^0, a_h^0) \tag{11}$$

Since $p(x_h^0 | x_{h-1}, a_{h-1}) \leq 1$, for any (x_{h-1}, a_{h-1}) at step $h-1$, we have:

$$\begin{aligned} 0 &\leq \gamma \left[\mathbb{E}_{x_h | x_{h-1}, a_{h-1}} \left[V_i(x_h; \mathbf{r}_1) \right] - \mathbb{E}_{x_h | x_{h-1}, a_{h-1}} \left[V_i(x_h; \mathbf{r}_2) \right] \right] & (12) \\ &\leq \gamma \left[r_1(x_h^0, a_h^0) - r_2(x_h^0, a_h^0) \right] & (13) \end{aligned}$$

Applying Lemma 1 and notice that all other rewards does not change, we have:

$$0 \leq V_i(x_{h-1}; \mathbf{r}_1) - V_i(x_{h-1}; \mathbf{r}_2) \leq \gamma \left[r_1(x_h^0, a_h^0) - r_2(x_h^0, a_h^0) \right] \tag{14}$$

We do this iteratively, and finally we have:

$$0 \leq V_i(s_i; \mathbf{r}_1) - V_i(s_i; \mathbf{r}_2) \leq \gamma^h \left[r_1(x_h^0, a_h^0) - r_2(x_h^0, a_h^0) \right] \tag{15}$$

We could show similar case when $r_1(x_h^0, a_h^0) < r_2(x_h^0, a_h^0)$, therefore, we have:

$$|V_i(s_i; \mathbf{r}_1) - V_i(s_i; \mathbf{r}_2)| \leq \gamma^h |r_1(x_h^0, a_h^0) - r_2(x_h^0, a_h^0)| \tag{16}$$

where $h = |x_h^0 - s_i|$ is the distance between s_i and x_h^0.

Now we consider general $\mathbf{r}_1 \neq \mathbf{r}_2$. We could design path $\{\mathbf{r}_t\}$ from $\mathbf{r}_1$ to $\mathbf{r}_2$ so that each time we only change one distinct reward entry. Therefore each (s, a) pairs happens only at most once and we have:

$$\begin{aligned} |V_i(s_i; \mathbf{r}_1) - V_i(s_i; \mathbf{r}_2)| &\leq \sum_t |V_i(s_i; \mathbf{r}_{t-1}) - V_i(s_i; \mathbf{r}_t)| & (17) \\ &\leq \sum_{x,a} \gamma^{|x - s_i|} |r_1(x, a) - r_2(x, a)| & (18) \end{aligned}$$

□

F.2 THM. 1

First we prove the following lemma:

Lemma 3. *For any reward assignments $\mathbf{r}_i$ for agent i for the optimization problem (Eqn. 1) and a local reward set $M_i^{\text{local}} \supseteq \{x : |x - s_i| \leq C\}$, if we construct $\tilde{\mathbf{r}}_i$ as follows:*

$$\tilde{r}_i(x, a) = \begin{cases} r_i(x, a) & x \in M_i^{\text{local}} \\ 0 & x \notin M_i^{\text{local}} \end{cases} \tag{19}$$

Then we have:

$$|V_i(s_i; \mathbf{r}_i) - V_i(s_i; \tilde{\mathbf{r}}_i)| \leq \gamma^C R_{\max} M \tag{20}$$

where M is the total number of sparse reward sites and $R_{\max}$ is the maximal reward that could be assigned at each reward site x while satisfying the constraint $\phi(r_1(x, a), r_2(x, a), \ldots, r_K(s, a)) \leq 0$.

Proof. By Lemma 2, we know that

$$\begin{aligned} |V_i(s_i; \mathbf{r}_i^*) - V_i(s_i; \tilde{\mathbf{r}}_i)| &\leq \sum_{x \notin S_i^{\text{local}}} \gamma^{|x - s_i|} |r_i^*(s, a) - \tilde{r}_i(s, a)| & (21) \\ &\leq \gamma^C \sum_{x \notin S_i^{\text{local}}} |r_i^*(s, a)| & (22) \\ &\leq \gamma^C R_{\max} M & (23) \end{aligned}$$

□

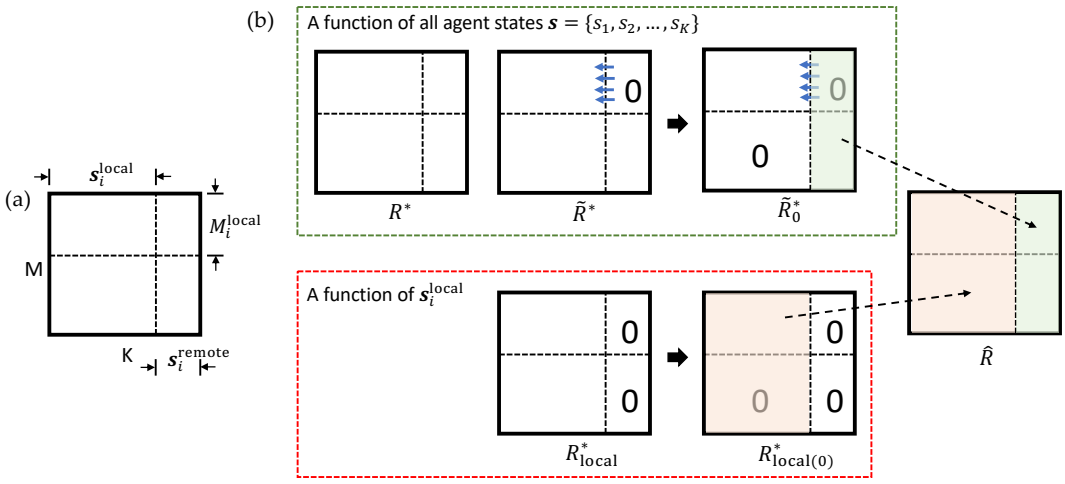

Figure 11: Different reward assignments.

Note that "sparse reward site" is important here, otherwise there could be exponential sites $x \notin S_i^{\text{local}}$ and Eqn. 23 becomes vacant.

Then we prove the theorem.

Proof. Given a constant C, for each agent i, we define the vicinity reward site $B_i(C) := \{x : |x - s_i| \leq C\}$.

Given agent i and its local "buddies" $\mathbf{s}_i^{\text{local}}$ (a subset of multiple agent indices), we construct the corresponding reward site set M_i^{local}:

$$M_i^{\text{local}} = \bigcup_{s_j \in \mathbf{s}_i^{\text{local}}} B_j(C) \tag{24}$$

Define the remote agents $\mathbf{s}_i^{\text{remote}} = \mathbf{s} \backslash \mathbf{s}_i^{\text{local}}$ as all agents that do not belong to $\mathbf{s}_i^{\text{local}}$.

Define the distance D between the M_i^{local} and $\mathbf{s}_i^{\text{remote}}$:

$$D = \min_{x \in M_i^{\text{local}}} \min_{s_j \in \mathbf{s}_i^{\text{remote}}} |x - s_j| \tag{25}$$

Intuitively, the larger D is, the more distant between relevant rewards sites from remote agents and the tighter the bound. There is a trade-off between C and D: the larger the vicinity, M_i^{local} expands and the smaller D is.

Given this setting, we then construct a few reward assignments (see Fig. 11), given the current agent states $\mathbf{s} = \{s_1, s_2, \ldots, s_K\}$. For brevity, we write $R[M, \mathbf{s}]$ to be the submatrix that relates to reward site M and agents set $\mathbf{s}$.

- The optimal solution R^* for Eqn. 1.
- The perturbed optimal solution $\tilde{R}^*$ by pushing the reward assignment of $[M_i^{\text{local}}, \mathbf{s}_i^{\text{remote}}]$ in R^* to $[M_i^{\text{local}}, \mathbf{s}_i^{\text{local}}]$.
- From $\tilde{R}^*$, we get $\tilde{R}_0^*$ by setting the region $[M_i^{\text{remote}}, \mathbf{s}_i^{\text{local}}]$ to be zero.
- The local optimal solution R^*_{local} that only depends on $\mathbf{s}_i^{\text{local}}$. This solution is obtained by setting $[:, \mathbf{s}_i^{\text{remote}}]$ to be zero and optimize Eqn. 1.
- From R^*_{local}, we get $R^*_{\text{local}(0)}$ by setting $[M_i^{\text{remote}}, \mathbf{s}_i^{\text{local}}]$ to be zero.

It is easy to show all these rewards assignment are feasible solutions to Eqn. 1. This is because if the original solution is feasible, then setting some reward assignment to be zero also yields a feasible solution, due to the property of the constraint ϕ.

For simplicity, we define J_{local} to be the partial objective that sums over $s_j \in \mathbf{s}_i^{\text{local}}$ and similarly for J_{remote}.

We could show the following relationship between these solutions:

$$J_{\text{remote}}(\tilde{R}^*) \geq J_{\text{remote}}(R^*) - \gamma^D R_{\max} MK \tag{26}$$

This is because each of this reward assignment move costs at most $\gamma^D R_{\max}$ by Lemma 2 and there are at most MK such movement.

On the other hand, for each $s_j \in \mathbf{s}_j^{\text{local}}$, since $M_i^{\text{local}} \supseteq B_j(C)$, from Lemma 3 we have:

$$V_j(R^*_{\text{local}(0)}) \geq V_j(R^*_{\text{local}}) - \gamma^C R_{\max} M \tag{27}$$

And similarly we have:

$$V_j(\tilde{R}^*_0) \geq V_j(\tilde{R}^*) - \gamma^C R_{\max} M \tag{28}$$

Now we construct a new solution $\hat{R}_i$ by combining $R^*_{\text{local}(0)}[:, \mathbf{s}_i^{\text{local}}]$ with $\tilde{\mathbf{r}}^*_0[:, \mathbf{s}_i^{\text{remote}}]$. This is still a feasible solution since in both $R^*_{\text{local}(0)}$ and $\tilde{R}^*_0$, their top-right and bottom-left sub-matrices are zero, and its objective is still good:

$$\begin{aligned}
J(\hat{R}) &= J_{\text{local}}(R^*_{\text{local}(0)}) + J_{\text{remote}}(\tilde{R}^*_0) && (29)\\
&\overset{①}{\geq} J_{\text{local}}(R^*_{\text{local}}) - \gamma^C R_{\max} MK + J_{\text{remote}}(\tilde{R}^*_0) && (30)\\
&\overset{②}{\geq} J_{\text{local}}(\tilde{R}^*) + J_{\text{remote}}(\tilde{R}^*_0) - \gamma^C R_{\max} MK && (31)\\
&\overset{③}{\geq} J_{\text{local}}(R^*) + J_{\text{remote}}(\tilde{R}^*_0) - \gamma^C R_{\max} MK && (32)\\
&\overset{④}{=} J_{\text{local}}(R^*) + J_{\text{remote}}(\tilde{R}^*) - \gamma^C R_{\max} MK && (33)\\
&\overset{⑤}{\geq} J_{\text{local}}(R^*) + J_{\text{remote}}(R^*) - R_{\max} MK(\gamma^C + \gamma^D) && (34)\\
&\overset{⑥}{=} J(R^*) - R_{\max} MK(\gamma^C + \gamma^D) && (35)
\end{aligned}$$

Note that ① is due to Eqn. 27, ② is due to the optimality of R^*_{local} (and looser constraints for R^*_{local}), ③ is due to the fact that $\tilde{R}^*$ is obtained by *adding* rewards released from $\mathbf{s}_i^{\text{remote}}$ to $\mathbf{s}_i^{\text{local}}$. ④ is due to the fact that $\tilde{R}^*_0$ and $\tilde{R}^*$ has the same remote components. ⑤ is due to Eqn. 26. ⑥ is by definition of J_{local} and J_{remote}.

Therefore we obtain $\hat{\mathbf{r}}_i = [\hat{R}]_i$ that only depends on $\mathbf{s}_i^{\text{local}}$. On the other hand, the solution $\hat{R}$ is close to optimal R^*, with gap $(\gamma^C + \gamma^D) R_{\max} MK$. □