A Architecture and hyperparameters

In this paper, we use lightweight network structures for the subtask encoder, subtask decoder, trajectory encoder and subtask policy. Specifically, the subtask encoder is a two-layer 64-dimensional fully-connected network and the output layer is followed by the tanh activation. Agents share a trajectory encoder to construct their ability representations. The trajectory encoder consists of three layers, a 64-dimensional fully-connected layer, followed by a 64-bit GRU, and followed by another 64-dimensional fully-connected layer. The new shared trajectory encoder only consists of a 64-dimensional fully-connected layer followed by a 64-bit GRU and the outputs will be fed into the subtask policy. The policy of each subtask is a fully-connected layer of $n\_actions$ dimensions, where $n\_actions$ is the number of actions. The parameters of the subtask policy are generated by the subtask encoder, which is also a fully-connected layer. We use the QMIX-style mixing network with the same setting as the original paper.

For all methods, we run 5 million timesteps on three *Super Hard* scenarios: `corridor`, `3s5z_vs_3s6z`, `6h_vs_8z` and 2 million timesteps on the other 11 scenarios. The replay buffer is a first-in-first-out queue with a max size of 5000 episodes. For every one episode data collected, we sample a batch of 32 episodes from the replay buffer for training. The decentralized policy is evaluated every $10k$ timesteps with 32 episodes. We use ϵ-greedy exploration. The ϵ anneals linearly from 1.0 to 0.05 over $50k$ timesteps and keeps constant for the rest of the learning. The training objective is optimized by RMSprop with a learning rate of 5×10^{-4}. All experiments are conducted on GeForce RTX 2080Ti GPU.

B Remaining results on the SMAC benchmark

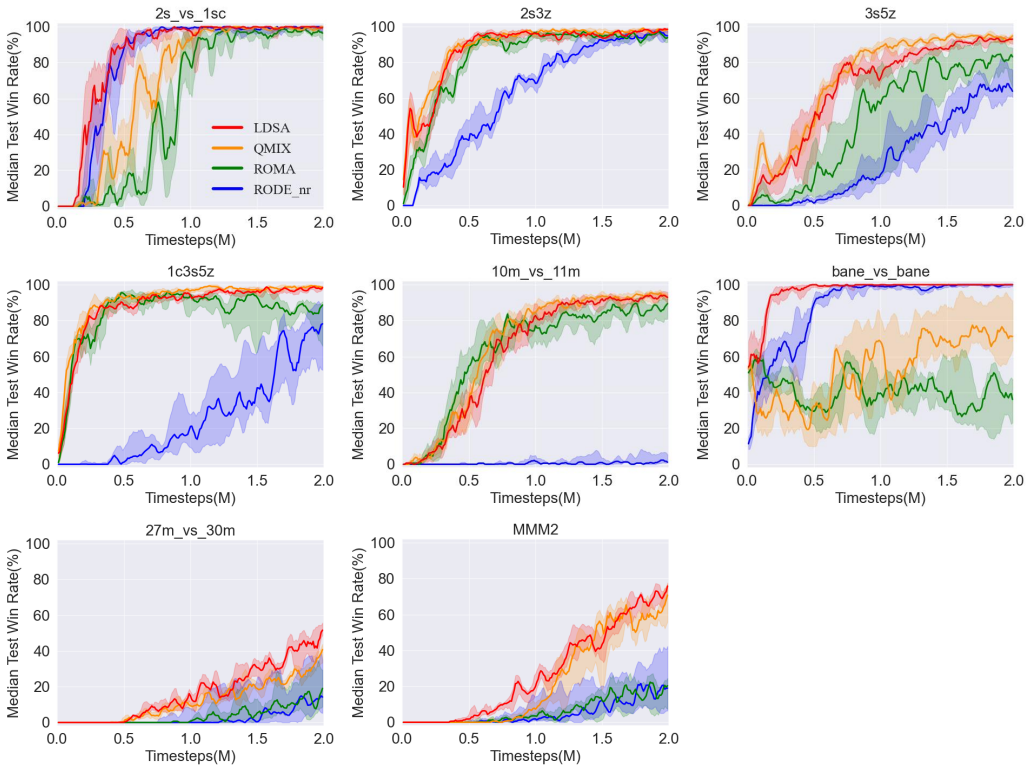

Figure 1: Comparison of our method against baselines on the remaining SMAC scenarios, including five *Easy* scenarios: `2s_vs_1sc`, `2s3z`, `3s5z`, `1c3s5z`, `10m_vs_11m` and one *Hard* scenario: `bane_vs_bane` and two *Super Hard* scenarios: `27m_vs_30m`, `MMM2`.

C Effect of the number of subtasks

The number of subtasks k is a core hyperparameter of LDSA. To study the effect of k, we test LDSA with different values of k on the SMAC scenario `corridor` as shown in Fig. 2. It can be observed that LDSA can obtain a better performance than QMIX when $k = 2$ and increasing the number of subtasks benefits the performance. But LDSA with more than 4 subtasks shows little improvement and learning more subtasks will lead to higher computational cost. Therefore, we set $k = 4$ throughout the work to balance the effectiveness and efficiency.

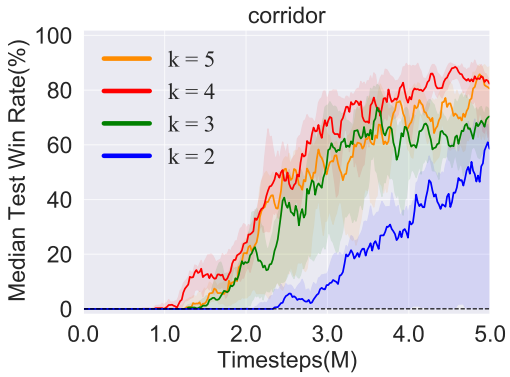

Figure 2: The performance of LDSA with different number of subtasks on the SMAC scenario `corridor`. The best performance of QMIX is shown as a dashed line.

D Comparison of LDSA with CDS on SMAC

In this section, we compare LDSA with CDS [1] on three SMAC scenarios: `5m_vs_6m`, `10m_vs_11m` and `27m_vs_30m` as shown in Fig. 3. We can see that LDSA performs better than CDS on all three scenarios. Moreover, CDS can't learn anything on the scenario `27m_vs_30m` that needs to control a large number of agents. This may be because CDS requires to learn one shared policy network and n non-shared policy networks (*i.e.*, a total of $n + 1$ policy networks) if we have n agents, which may lead to high training complexity when the multi-agent task contains a large number of agents. Our method only needs to learn k subtask policies, where $k \ll n$ when controlling a large number of agents. Therefore, compared with CDS, LDSA can achieve a better balance between the training complexity and the diversity of agent behavior.

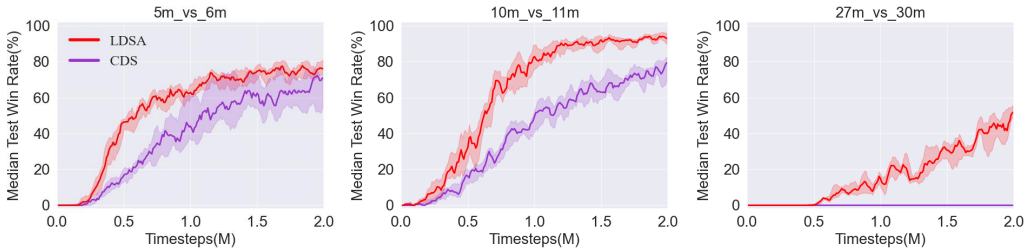

Figure 3: Comparison of LDSA with CDS on three SMAC scenarios: `5m_vs_6m`, `10m_vs_11m` and `27m_vs_30m`.

E Performance on GRF

To validate the benefits of LDSA on more multi-agent tasks, we evaluate our method on two challenging Google Research Football (GRF) [2] academy scenarios `academy_3_vs_1_with_keeper` and `academy_counterattack_easy`. On both scenarios, we can control more players against two enemy players. We can solve these two tasks similarly by learning two main different subtasks. One

is to dribble the ball to the top side of the field and attract the attention of the enemy players to create an enemy vacancy in the down side of the field. The other is to run to the down side of the field and wait for a long pass from teammates on the top side, and then shoot. We show the median and 25-75% percentiles of the test score reward across five random seeds in Fig. 4. LDSA could also achieve better performance than baselines on GRF, which demonstrates the effectiveness of LDSA on various multi-agent tasks.

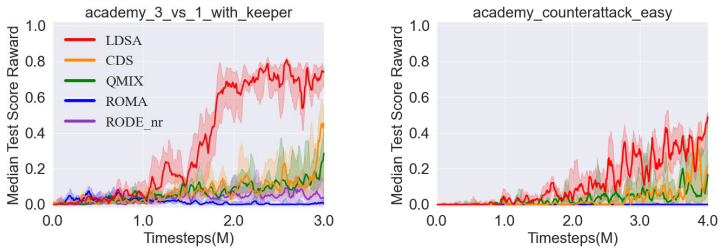

Figure 4: Comparison of our method against baselines on GRF.

F Additional ablations

In this section, We conduct two more ablations for LDSA, named LDSA_DireProb and LDSA_Mix, respectively. LDSA_DireProb means to directly estimate the subtask selection distribution from agents' observed trajectories. LDSA_Mix represents that each agent uses a mixture of subtask policies as opposed to sampling one subtask policy. We compare these two ablations with LDSA on the SMAC *Super Hard* scenario `corridor` as shown in Fig. 5. We can observe that the performance of LDSA_Mix is lower than that of LDSA, which indicates that it's better for each agent to learn one subtask rather than a mixture of all subtasks at each timestep. Besides, the comparison between LDSA and LDSA_DireProb highlights the importance of computing the subtask selection distribution based on similarity of agent trajectories and subtask representations. In other words, if we have explicit subtask representation and the subtask policy depend on its representation, it's hard to learn to select the subtask if we have no information about the subtask. These two more ablations further confirm the effectiveness of LDSA.

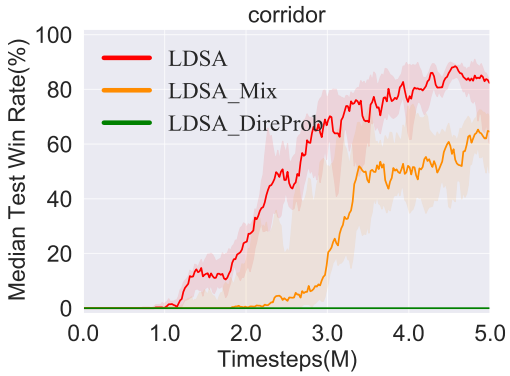

Figure 5: Additional ablation studies regarding components of LDSA on the SMAC scenario `corridor`.

References

[1] L. Chenghao, T. Wang, C. Wu, Q. Zhao, J. Yang, and C. Zhang, "Celebrating diversity in shared multi-agent reinforcement learning," in *Proceedings of the Neural Information Processing Systems*, vol. 34, 2021.

[2] K. Kurach, A. Raichuk, P. Stańczyk, M. Zając, O. Bachem, L. Espeholt, C. Riquelme, D. Vincent, M. Michalski, O. Bousquet *et al.*, "Google research football: A novel reinforcement learning environment," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 4501–4510.