Solving Continual Combinatorial Selection via Deep Reinforcement Learning

Hyungseok Song[1] , **Hyeryung Jang** [2] , **Hai Tran Hong**[1] , **Seeun Yun**[1] ,
Donggyu Yun[3] , **Hyoju Chung**[3] , **Yung Yi**[1] ,
[1]First Affiliation
[2]Second Affiliation
[3]Third Affiliation
{first, second}@example.com, third@other.example.com, fourth@example.com

Abstract

We consider the Markov Decision Process (MDP) of selecting a subset of items at each step, termed the Select-MDP (S-MDP). The large state and action spaces of S-MDPs make them intractable to solve with typical reinforcement learning (RL) algorithms, especially when it comes to real-world problems where the number of items is often huge. In this paper, we present a deep RL algorithm to solve this issue by adopting the following key ideas. First, we convert the original S-MDP into an *Iterative Select-MDP (IS-MDP)*, which is equivalent to the S-MDP in terms of optimal actions. IS-MDP decomposes a joint action of selecting K items simultaneously into K iterative selections resulting in the decrease of actions at the expense of an exponential increase of states. Second, we overcome this state space explosion by exploiting a special property that exists uniquely in IS-MDPs - the *equi-invariance*. Equi-invariance paves way for the design of our significantly simplified K-cascaded deep Q-networks based on a method named *progressive-parameter sharing*. Various experiments demonstrate that our approach works well even when the item space is large and that it scales to environments with item spaces different from those used in training.

1 Introduction

Imagine yourself managing a football team in a league of many matches. Your goal is to maximize the total number of winning matches during the league. For each match, you decide a lineup (*action*: $\tilde{a}$) by selecting K players among N candidates (*item*) to participate in it and allocating one of C positions (*command*) to each of them, with possible duplication. You can observe a collection (*state*: $\tilde{s}$) of the current status (*information*) of each candidate player. During the match, you cannot supervise anymore until you receive the result (*reward*: $\tilde{r}$), as well as the changed collection of the status (*next state*: $\tilde{s}'$) of N players. In order to win the long league, you should pick a proper combination of the selected players and their positions to achieve not only a myopic result of the following match but also to consider a long-term plan such as the rotation of the members. We model an MDP for these kinds of problems, termed *Select-MDP* (S-MDP), where an agent needs to make combinatorial selections sequentially.

There are many applications that can be formulated as an S-MDP including recommendation systems [Ricci *et al.*, 2015; Zheng *et al.*, 2018], contextual combinatorial semi-bandits [Qin *et al.*, 2014; Li *et al.*, 2016], mobile network scheduling [Kushner and Whiting, 2004], and fully-cooperative multi-agent systems controlled by a centralized agent [Usunier *et al.*, 2017] (when $N = K$). However, learning a good policy is challenging because the state and action spaces increase exponentially in K and N. For example, our experiment shows that the vanilla DQN [Mnih *et al.*, 2015] proposed to tackle the large state space issue fails to learn the Q-function in our test environment of $N = 50$, even for the simplest case of $C = 1$, $K = 1$. This motivates the research on a scalable RL algorithm that produces a good policy for tasks modeled by an S-MDP.

In this paper, we present a novel DQN-based RL algorithm for S-MDPs by adopting a synergic combination of the following two design ideas:

D1. For a given S-MDP, we convert it into a divided but equivalent one, called *Iterative Select-MDP* (IS-MDP), where the agent iteratively selects an (item, command) pair one by one during K steps rather than K items at once. This transformation significantly relieves the complexity of the joint action space per state in S-MDP; the agent only needs to evaluate KNC actions in IS-MDP, whereas it should compute $\binom{N}{K}C^K$ actions for each step in S-MDP. We design K-cascaded deep Q-networks for IS-MDP, where each Q-network selects an item with an assigned command respectively while considering the selections by previous cascaded networks.

D2. Although we significantly reduce per-state action space in IS-MDP, the state space is still large as N or K grows. To have scalable and fast training, we consider two different parameter sharing methods: I-sharing and U-sharing. I-sharing is a weight sharing method for each cascaded Q-network to handle the complexity by N. This is done by exploiting a special symmetry in IS-MDP called *equi-invariance* where the order of items does not matter. We further simplify those cascaded Q-networks by sharing weight parameters among them, which reduces the training complexity for large K. In pactice, we propose to use a mixture of I- and U-sharing, which we call P-sharing (progressive sharing), by starting from a single parameter set as in U-sharing and then progressively increasing the number of parameter sets, approaching to that of I-sharing.

The superiority of our ideas is discussed and evaluated in two ways. First, despite the drastic parameter reduction, we claim that I-sharing does not hurt the expressive power too much by proving (i) relative local optimality and (ii) universality of I-sharing. Note that this analytical result is not limited to a Q-function approximator in RL, but is also applied to any neural network with parameter sharing in other contexts such as supervised learning. Second, we evaluate our approach on two self-designed S-MDP environments (circle selection and selective predator-prey) and observe a significantly high performance improvement, especially with large N (e.g., $N = 200$), over other baselines. Moreover, the trained parameters can generalize to other environments of much larger item sizes without additional training, where we use the trained parameters in $N = 50$ for those in $N = 200$.

1.1 Related Work

Combinatorial Optimization via RL Recent works on deep RL have been solving NP-hard combinatorial optimization problems on graphs [Dai *et al.*, 2017], Traveling Salesman problems [Kool *et al.*, 2019], and recommendation systems [Chen *et al.*, 2018; Deudon *et al.*, 2018]. In many works for combinatorial optimization problems, they do not consider the future state after selecting a combination of K items and some other commands. [Chen *et al.*, 2018] suggests similar cascaded Q-networks without efficient weight sharing which is crucial in handling large dimensional items. [Usunier *et al.*, 2017] suggests a centralized MARL algorithm where the agent randomly selects an item first and then considers the command. Independent Deep Q-network (IDQN) [Tampuu *et al.*, 2017] is an MARL algorithm where each item independently chooses its command using its Q-network. To summarize, our contribution is to extend and integrate those combinatorial optimization problems successfully and to provide a scalable RL algorithm using weight shared Q-networks.

Parameter Sharing on Neural Networks and Analysis Parameter shared neural networks have been studied on various structured data domains such as graphs [Kipf and Welling, 2017] and sets. These networks do not only save significant memory and computational cost but also perform usually better than non-parameter shared networks. For the case of set-structured data, there are two major categories: equivariant [Ravanbakhsh *et al.*, 2017a; Jason and Devon R Graham, 2018] and invariant networks [Qi *et al.*, 2017; Zaheer *et al.*, 2017; Maron *et al.*, 2019]. In this paper, we develop a parameter shared network (I-sharing) which contains both permutation equivariant and invariant properties. Empirical successes of parameter sharing have led many works to delve into its mathematical properties. [Qi *et al.*, 2017; Zaheer *et al.*, 2017; Maron *et al.*, 2019] show the universality of invariant networks for various symmetries. As for equivariant networks, a relatively small number of works analyze their performance. [Ravanbakhsh *et al.*, 2017b; Zaheer *et al.*, 2017; Jason and Devon R Graham, 2018] find necessary and sufficient conditions of equivariant linear layers. [Yarotsky, 2018]

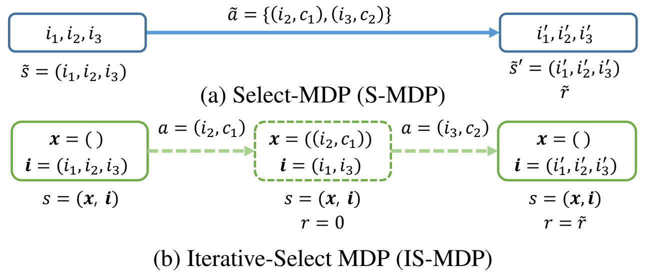

$\tilde{a} = \{(i_2, c_1), (i_3, c_2)\}$
i_1, i_2, i_3
i'_1, i'_2, i'_3
$\tilde{s} = (i_1, i_2, i_3)$
$\tilde{s}' = (i'_1, i'_2, i'_3)$
$\tilde{r}$
(a) Select-MDP (S-MDP)
$a = (i_2, c_1)$
$a = (i_3, c_2)$
$\boldsymbol{x} = ()$
$\boldsymbol{i} = (i_1, i_2, i_3)$
$\boldsymbol{x} = ((i_2, c_1))$
$\boldsymbol{i} = (i_1, i_3)$
$\boldsymbol{x} = ()$
$\boldsymbol{i} = (i'_1, i'_2, i'_3)$
$s = (\boldsymbol{x}, \boldsymbol{i})$
$s = (\boldsymbol{x}, \boldsymbol{i})$
$r = 0$
$s = (\boldsymbol{x}, \boldsymbol{i})$
$r = \tilde{r}$
(b) Iterative-Select MDP (IS-MDP)

Figure 1: Example of an S-MDP and its equivalent IS-MDP for $N = 3$ and $K = 2$.

designs a universal equivariant network based on polynomial layers. However, their polynomial layers are different from widely used linear layers. In our paper, we prove two theorems which mathematically guarantee the performance of our permutation equi-invariant networks in different ways. Both theorems can be applied to other similar related works.

2 Preliminary

2.1 Iterative Select-MDP (IS-MDP)

Given an S-MDP, we formally describe an IS-MDP as a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ that makes a selection of K items and corresponding commands in an S-MDP through K consecutive selections. Fig. 1 shows an example of the conversion from an S-MDP to its equivalent IS-MDP. In IS-MDP, given a tuple of the N-item information $(i_1, \dots, i_N)$, with $i_n \in \mathcal{I}$ being the information of the item n, the agent selects one item i_n and assigns a command $c \in \mathcal{C}$ at every 'phase' k for $0 \le k < K$. After K phases, it forms a joint selection of K items and commands, and a probabilistic transition of the N-item information and the associated reward are given.

To elaborate, at each phase k, the agent observes a state $s = ((x_1, \dots, x_k), (i_1, \dots, i_{N-k})) \in \mathcal{S}_k$ which consists of a set of k pairs of information and command which are selected in prior phases, denoted as $\boldsymbol{x} = (x_1, \dots, x_k)$, with $x_k \in \mathcal{I} \times \mathcal{C}$ being a pair selected in the kth phase, and a tuple of information of the unselected items up to phase k, denoted as $\boldsymbol{i} = (i_1, \dots, i_{N-k})$. From the observation $s \in \mathcal{S}_k$ at phase k, the agent selects an item n among the $N - k$ unselected items and assigns a command c, i.e., a feasible action space for state s is given by $\mathcal{A}(s) := \{(n, c) \mid n \in \{1, \dots, N - k\}, c \in \mathcal{C}\}$, where (n, c) represents a selection (i_n, c). As a result, the state and action spaces of an IS-MDP are given by $\mathcal{S} = \bigcup_{0 \le k < K} \mathcal{S}_k$ and $\mathcal{A} = \bigcup_{s \in \mathcal{S}} \mathcal{A}(s)$, respectively. We note that any state $\tilde{s} = (i_1, \dots, i_N)$ in an S-MDP belongs to $\mathcal{S}_0$, i.e., the 0th phase. In an IS-MDP, action $a = (n, c) \in \mathcal{A}(s)$ for state $s = (\boldsymbol{x}, \boldsymbol{i}) \in \mathcal{S}_k$ results in the next state $s' = (\boldsymbol{x} + (i_n, c), \boldsymbol{i} - i_n) \in \mathcal{S}_{k+1}$[1] and a reward r, which are characterized by the transition probability $\tilde{\mathcal{P}}$ of S-MDP as

$$\begin{aligned} k < K - 1, &\quad \mathcal{P}(s', 0 \mid s, a) \equiv 1, \\ k = K - 1, &\quad \mathcal{P}(s', \tilde{r} \mid s, a) \equiv \tilde{\mathcal{P}}(\tilde{s}', \tilde{r} \mid \tilde{s}, \tilde{a}). \end{aligned} \quad (1)$$

The decomposition of joint action in S-MDPs (i.e., selecting K items at once) into K consecutive selections in IS-MDPs has

[1]We use $+, -$ as $\boldsymbol{x} + x := (x_1, \cdots, x_k, x)$ and $\boldsymbol{i} - i_n := (i_k, \cdots, i_{n-1}, i_{n+1}, \cdots, i_N)$.

equivalence in terms of the optimal policy [Maes *et al.*, 2009]. Important advantage from the decomposition is that IS-MDPs have action space $\mathcal{A}$ of size KNC while the action space of S-MDPs is $\binom{N}{K}C^K$. However, two challenges are added: (i) the state space $\mathcal{S}$ of IS-MDPs is exponentially increasing as K grows, and (ii) the dimensions of state space $\mathcal{S}_k$ and action space $\{\mathcal{A}(s)\}_{s\in\mathcal{S}_k}$ are different from phase to phase. We aim at proposing a method to address (i) and (ii) in Sec. 3.

2.2 Deep Q-network (DQN)

We provide a background of the DQN [Mnih *et al.*, 2015], one of the standard deep RL algorithms, whose key ideas such as the target network and replay buffer will also be used in our proposed method. The goal of RL is to learn an optimal policy $\pi^\star(a|s) : \mathcal{S}\times\mathcal{A}\mapsto[0,1]$ that maximizes the expected discounted return. We denote the optimal action-value functions (Q-function) under the optimal policy $\pi^\star$ by $Q^\star(s,a)$. The deep Q-network (DQN) parameterizes and approximates the optimal Q-function $Q^\star(s,a)$ using the so-called Q-network $Q(s,a;\omega)$, i.e., a deep neural network with a weight parameter vector ω. In DQN, the parameter ω is learned by sampling minibatches of experience (s,a,r,s') from the replay buffer and using the following loss function:

$$l(\omega)=\Big(Q(s,a\,;\omega)-(r+\gamma\max_{a'\in\mathcal{A}(s')}Q(s',a'\,;\omega'))\Big)^2 \quad (2)$$

where ω' is the target parameter which follows the main parameter ω slowly. It is common to approximate $Q(s;\omega):\mathcal{S}\mapsto\mathbb{R}^{|\mathcal{A}(s)|}$ rather than $Q(s,a;\omega)$ using a neural network so that all action values can be easily computed at once.

3 Methods

In this section, we present a symmetric property of IS-MDP, which is referred to as *Equi-Invariance* (EI), and propose an efficient RL algorithm to solve IS-MDP by constructing K cascaded Q-networks with two-levels of parameter sharing.

3.1 IS-MDP: Equi-Invariance

As mentioned in Sec. 2.1, a state $s=(\boldsymbol{x},\boldsymbol{i})$ at phase k includes two sets $\boldsymbol{x}$ and $\boldsymbol{i}$ of observations, so that we have some permutation properties related to the ordering of elements in each set. To elaborate, we denote as $\sigma_s=(\sigma_x,\sigma_i)\in\boldsymbol{S}_k\times\boldsymbol{S}_{N-k}$ a permutation of a state s at phase k, which is defined as

$$\sigma_s(s):=(\sigma_x(\boldsymbol{x}),\sigma_i(\boldsymbol{i})), \quad (3)$$

where $\boldsymbol{S}_k$ is a group of permutations of a set with k elements. Under the optimal policy, the equivalent states s and $\sigma_s(s)$ should be treated equally. If the action $a=(n,c)\in\mathcal{A}(s)$ is the best action for s, then for state $\sigma_s(s)$, an optimal policy should choose a proper associated action $\sigma_i(a):=(\sigma_i(n),c)$. As a result, we have $\forall s\in\mathcal{S},\forall a\in\mathcal{A}(s)$,

$$Q^\star(s,a)=Q^\star(\sigma_s(s),\sigma_i(a)). \quad (4)$$

Focusing on Q-value function $Q^\star(s)=[Q^\star(s,a)]_{a\in\mathcal{A}(s)}$, as discussed in Sec. 2.2, a permutation $\sigma_s=(\sigma_x,\sigma_i)$ of a state s permutes the output of the function $Q^\star(s)$ according to the permutation σ_i. In other words, a state s and the permutation thereof, $\sigma_s(s)$, have *equi-invariant* optimal Q-value function $Q^\star(s)$. This is stated in the following proposition which is a rewritten form of (4).

Proposition 1 (Equi-Invariance of IS-MDP)**.** *In IS-MDP, the optimal Q-function $Q^\star(s)$ of any state $s=(\boldsymbol{x},\boldsymbol{i})\in\mathcal{S}$ is invariant to the permutation of a set $\boldsymbol{x}$ and equivariant to the permutation of a set $\boldsymbol{i}$, i.e. for any permutation $\sigma_s=(\sigma_x,\sigma_i)$,*

$$Q^\star(\sigma_s(s))=\sigma_i(Q^\star(s)). \quad (5)$$

As we will discuss later, this EI property in (5) plays a critical role in reducing state and action spaces by considering (s,a) pairs and permutations thereof to be the same. We follow the idea in [Zinkevich and Balch, 2001] to prove Proposition 1.

3.2 Iterative Select Q-learning (ISQ)

Cascaded Deep Q-networks As mentioned in Sec. 2.1, the dimensions of state and action spaces differ over phases. In particular, as the phase k progresses, the set $\boldsymbol{x}$ of the state increases while the set $\boldsymbol{i}$ and the action space $\mathcal{A}(s)$ decrease. Recall that the action space of state $s\in\mathcal{S}_k$ is $\mathcal{A}(s)=\{(n,c)\mid n\in\{1,\dots,N-k\},c\in\mathcal{C}\}$. Then, Q-value function at each phase k, denoted as $Q_k(s)=[Q(s,a)]_{a\in\mathcal{A}(s)}$ for $s\in\mathcal{S}_k$, is characterized by a mapping from a state space $\mathcal{S}_k$ to $\mathbb{R}^{(N-k)\times C}$, where the (n,c)-th output element corresponds to the value $Q(s,a)$ of $a=(n,c)\in\mathcal{A}(s)$.

To solve IS-MDP using a DQN-based scheme, we construct K deep Q-networks that are cascaded, where the kth Q-network, denoted as $Q_k(s;\omega_k)$, approximates the Q-value function $Q_k(s)$ with a learnable parameter vector ω_k. We denote by $\boldsymbol{\omega}=\{\omega_k\}_{0\le k<K}$ and $\boldsymbol{\omega}'=\{\omega'_k\}_{0\le k<K}$ the collections of the main and target weight vectors for all K-cascaded Q-networks, respectively. With these K-cascaded Q-networks, DQN-based scheme can be applied to each Q-network $Q_k(s;\omega_k)$ for $0\le k<K$ using the associated loss function as in (2) with $\omega=\omega_k$ and $\omega'=\omega'_{k+1}$ (since $s'\in\mathcal{S}_{k+1}$), which we name *Iterative Select Q-learning (ISQ)*.

Clearly, a naive ISQ algorithm would have training challenges due to the large-scale of N and K since (i) number of parameters in each network ω_k increases as N increases and (ii) size of the parameter set $\boldsymbol{\omega}$ also increases as K increases. To overcome these, we propose parameter sharing ideas which are described next.

Intra Parameter Sharing (I-sharing) To overcome the parameter explosion for large N in each Q-network, we propose a parameter sharing scheme, called *intra parameter sharing* (I-sharing). Focusing on the kth Q-network without loss of generality, the Q-network with I-sharing has a reduced parameter vector θ_k[2], yet it satisfies the EI property in (5), as discussed shortly.

The Q-network with I-sharing $Q_k(\cdot;\theta_k)$ is a multi-layered neural network constructed by stacking two types of parameter-shared layers: ϕ_k and ψ_k. As illustrated in Fig. 2, where the same colored and dashed weights are tied together, the layer ϕ_k is designed to preserve an *equivariance* of the permutation $\sigma_s=(\sigma_x,\sigma_i)\in\boldsymbol{S}_k\times\boldsymbol{S}_{N-k}$, while the layer ψ_k is designed to satisfy *invariance* of σ_x as well as *equivariance* of σ_i, i.e.,

$$\phi_k(\sigma_s(\boldsymbol{x},\boldsymbol{i}))=\sigma_s(\phi_k(\boldsymbol{x},\boldsymbol{i})),$$
$$\psi_k(\sigma_s(\boldsymbol{x},\boldsymbol{i}))=\sigma_i(\psi_k(\boldsymbol{x},\boldsymbol{i})).$$

[2]To distinguish the parameters of Q-networks with and without I-sharing, we use notations θ_k and ω_k for each case, respectively.

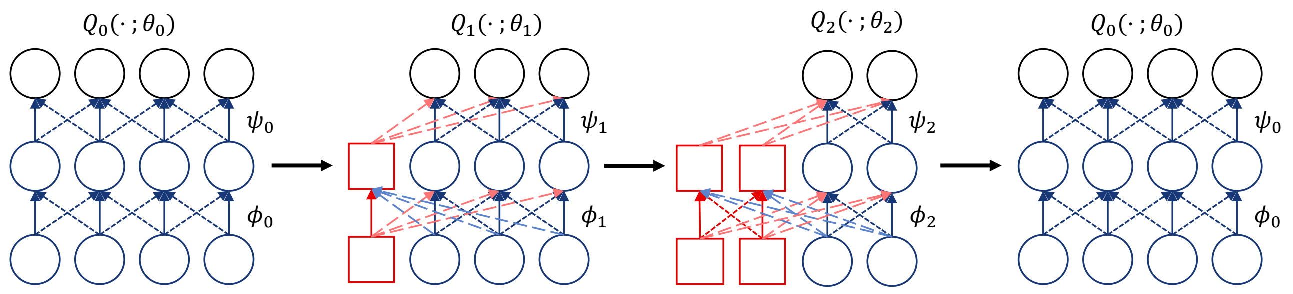


Figure 2: A simple example of the parameter-shared Q-networks $Q_k(\cdot\,;\theta_k)$ when $K = 3, N = 4, |\mathcal{C}|= 1$. Red and blue colored nodes represent the nodes equivariant to the selected items $\boldsymbol{x}$ and the unselected items $\boldsymbol{i}$ respectively. Each black node represents the Q value for selecting the corresponding (item, command) pair.

Then, we construct the Q-network with I-sharing $Q_k(\cdot;\theta_k)$ by first stacking multiple layers of ϕ_k followed by a single layer of ψ_k as

$$Q_k(s;\theta_k) := \psi_k \circ \phi_k \circ \cdots \circ \phi_k(s),$$

where θ_k is properly set to have tied values. Since composition of the permutation equivariant/invariant layers preserves the permutation properties, we obtain the following EI property

$$Q_k(\sigma_s(\boldsymbol{x}, \boldsymbol{i}); \theta_k) = \sigma_i(Q_k(\boldsymbol{x}, \boldsymbol{i}; \theta_k)).$$

ISQ algorithm with I-sharing, termed ISQ-I, achieves a significant reduction of the number of parameters from $|\boldsymbol{\omega}|= O(N^2K)$ to $|\boldsymbol{\theta}|= O(K)$, where $\boldsymbol{\theta} = \{\theta_k\}_{0\le k<K}$ is the collection of the parameters. We refer the readers to our technical report [Report, 2019] for a more mathematical description of I-sharing.

Unified Parameter Sharing (U-sharing) We propose an another-level of weight sharing method for ISQ, called *unified parameter sharing* (U-sharing). We observe that each I-shared Q-network $Q_k(\cdot\,;\theta_k)$ has a fixed number of parameters regardless of phase k. This is well described in Fig. 2, where the number of different edges are the same in Q_1 and Q_2. From this observation, we additionally share θ_k among the different Q-networks Q_k, i.e. $\theta_0 = \cdots = \theta_{K-1}$. U-sharing enables the reduction of the number of weights from $O(K)$ for $\boldsymbol{\theta}$ to $O(1)$ for $\theta_0 = \cdots = \theta_{K-1}$. Our intuition for U-sharing is that since the order of the selected items does not affect the transition of S-MDP, the criteria for selecting items is similar in every phase. This implies that the weight vectors θ_k may also have similar values. However, too aggressive sharing such as sharing all the weights may experience significantly reduced expressive power.

Progressive Parameter Sharing (P-sharing) To take the advantages of both I- and U-sharing, we propose a combined method called *progressive parameter sharing* (P-sharing). In P-sharing, we start with a single parameter set (as in U-sharing) and then progressively double the number of sets until it reaches K (the same as I-sharing). The Q-networks with nearby phases (Q_k and Q_{k+1}) tend to share a parameter set longer as visualized in Fig. 3, which we believe is because they have a similar criterion. In the early unstable stage of the learning, the Q-networks are trained sample-efficiently as they exploit the advantages of U-sharing. As the training continues, the Q-networks are able to be trained more elaborately, with more accurate expressive power, by increasing the number of

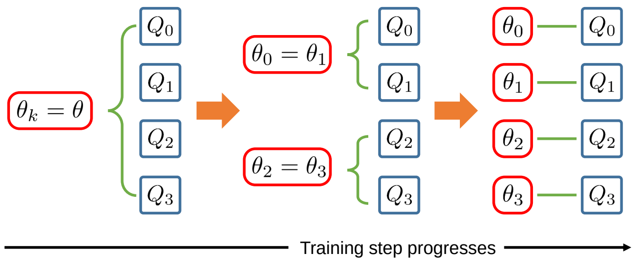


Figure 3: Illustration of P-sharing for $K = 4$. In the beginning, all Q-networks share the same weights. As the training progresses, we double the number of parameter sets until each Q-network Q_k is trained with its own parameter vectors θ_k.

parameter sets. In P-sharing, the number of the total weight parameters ranges from $O(1)$ to $O(K)$ during training.

4 Intra-Sharing: Optimality and Universal Approximation

One may naturally raise the question of whether the I-shared Q-network $Q_k(s\,;\theta_k) : \mathcal{S}_k \to \mathbb{R}^{|\mathcal{A}(s)|}$ has enough expressive power to represent the optimal Q-function $Q^\star_k(s) : \mathcal{S}_k \to \mathbb{R}^{|\mathcal{A}(s)|}$ of the IS-MDP despite the large reduction in the number of the parameters from $O(N^2K)$ to $O(K)$. In this section, we present two theorems that show $Q_k(s\,;\theta_k)$ has enough expressive power to approximate $Q^\star_k(s)$ with the EI property in (5). Theorem 1 states how I-sharing affects local optimality and Theorem 2 states whether the network still satisfies the universal approximation even with the equi-invariance property. Due to space constraint, we present the proof of the theorems in the technical report [Report, 2019]. We comment that both theorems can be directly applied to other similar weight shared neural networks, e.g., [Qi *et al.*, 2017; Zaheer *et al.*, 2017; Ravanbakhsh *et al.*, 2017b]. For presentational convenience, we denote $Q^\star_k(s)$ as $Q^\star(s)$, $Q_k(s\,;\omega_k)$ as $Q_\omega(s)$, and $Q_k(s\,;\theta_k)$ as $Q_\theta(s)$.

Relative Local Optimality We compare the expressive power of I-shared Q-network Q_θ and vanilla Q-network Q_ω of the same structure when approximating a function $Q^\star$ satisfies the EI property. Let Θ and Ω denote weight vector spaces for Q_θ and Q_ω, respectively. Since both Q_ω and Q_θ have the same network sructure, we can define a projection mapping $\omega : \Theta \to \Omega$ such that $Q_{\omega(\theta)} \equiv Q_\theta$ for any θ. Now, we introduce a loss surface function $l_\Omega(\omega)$ of the weight parameter vector ω:

$$l_\Omega(\omega) := \sum_{s \in B} |Q_\omega(s) - Q^\star(s)|^2,$$

where $B \subset \mathcal{S}_k$ is a batch of state samples at phase k and $Q^\star(s)$ implies the true Q-values to be approximated. Note that this loss surface l_Ω is different from the loss function of DQN in (2). However, from the EI property in $Q^\star(s)$, we can augment additional true state samples and the true Q-values by using equivalent states for all $\sigma_s \in \boldsymbol{S}_k \times \boldsymbol{S}_{N-k}$,

$$L_\Omega(\omega) := \sum_{\sigma_s \in \boldsymbol{S}_k \times \boldsymbol{S}_{N-k}} \left(\sum_{s \in B} |Q_\omega(\sigma_s(s)) - Q^\star(\sigma_s(s))|^2 \right).$$

We denote the loss surface $L_\Theta(\theta) := L_\Omega(\omega(\theta))$ in the weight shared parameter space Θ.

Theorem 1 (Relative Local Optimality). *If $\theta^\star \in \Theta$ is a local optimal parameter vector of the loss surface $L_\Theta(\theta)$, then the projected parameter $\omega(\theta^\star) \in \Omega$ is also the local optimal point of $L_\Omega(\omega)$.*

It is notoriously hard to find a local optimal point by using gradient descent methods because of many saddle points in high dimensional deep neural networks [Dauphin *et al.*, 2014]. However, we are able to efficiently seek for a local optimal parameter $\theta^\star$ on the smaller dimensional space Θ, rather than exploring Ω. The quality of the searched local optimal parameters $\omega(\theta^\star)$ is reported to be reasonable that most of the local optimal parameters give nearly optimal performance in high dimensional neural networks [Dauphin *et al.*, 2014; Kawaguchi, 2016; Laurent and Brecht, 2018] To summarize, Theorem 1 implies that Q_θ has similar expressive power to Q_ω if both have the same architecture.

Universal Approximation We now present a result related to the universality of $Q_\theta(s)$ when it approximates $Q^\star(s)$.

Theorem 2 (Universal Approximation). *Let $Q^\star : \mathcal{S}_k \to \mathbb{R}^{(N-k)\times C}$ satisfies EI property. If the domain spaces $\mathcal{I}$ and $\mathcal{C}$ are compact, for any $\epsilon > 0$, there exists a 4-layered I-shared neural network $Q_\theta : \mathcal{S}_k \to \mathbb{R}^{(N-k)\times C}$ with a finite number of neurons, which satisfies*

$$\forall s \in \mathcal{S}_k, \quad |Q^\star(s) - Q_\theta(s)| < \epsilon.$$

Both Theorems 1 and 2 represent the expressive power of the I-shared neural network for approximating an equi-invariant function. However, they differ in the sense that Theorem 1 directly compares the expressive power of the I-shared network to the network without parameter sharing, whereas Theorem 2 states the potential power of the I-shared network that any function f with EI property allows good approximation as the number of nodes in the hidden layers sufficiently increase.

5 Simulations

5.1 Environments and Tested Algorithms

Circle Selection (CS) In Circle Selection (CS) task, there are N selectable and U unselectable circles, where each circle is randomly moving and its radius increases with random noise. The agent observes positions and radius values of all the circles as a state, selects K circles among N selectable ones, and chooses 1 out of the 5 commands: moves *up, down, left, right*, or *stay*. Then, the agent receives a negative or zero reward if the selected circles overlap with unselectable or other selected circles, respectively; otherwise, it can receive a positive reward. The amount of reward is related to a summation of the selected circles' area. All selected circles and any overlapping unselectable circle are replaced by new circles, which are initialized at random locations with small initial radius. Therefore, the agent needs to avoid the overlaps by carefully choosing circles and their commands to move.

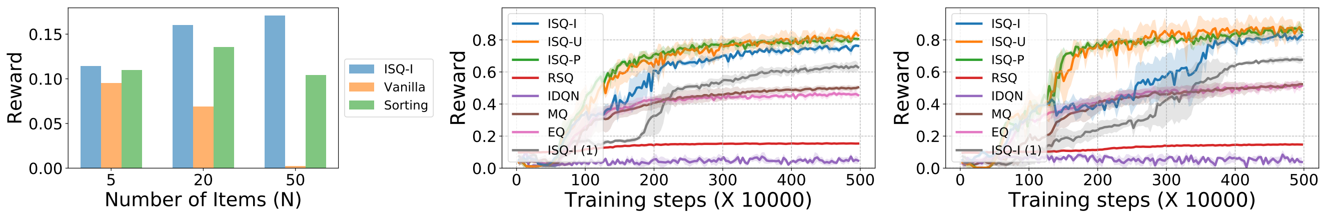
Selective Predator-Prey (PP) In this task, multiple predators capture randomly moving preys. The agent observes the positions of all the predators and preys, selects K predators, and assigns the commands as in the CS task. Only selected predators can move according to the assigned command and capture the preys. The number of preys caught by the predators is given as a reward, where a prey is caught if and only if more than two predators catch the prey simultaneously.

Tested Algorithms and Setup We compare the three variants of ISQ: ISQ-I, ISQ-U, ISQ-P with three DQN-based schemes: (i) a vanilla DQN [Mnih *et al.*, 2015], (ii) a sorting DQN that reduces the state space by sorting the order of items based on a pre-defined rule, and (iii) a myopic DQN which learns to maximize the instantaneous reward for the current step, but follows all other ideas of ISQ. We also consider three other baselines motivated by value-based MARL algorithms in [Tampuu *et al.*, 2017; Usunier *et al.*, 2017; Chen *et al.*, 2018]: Independent DQN (IDQN), Random-Select DQN (RSQ), and Element-wise DQN (EQ). In IDQN, each item observes the whole state and has its own Q-function with action space equals to $\mathcal{C}$. In RSQ, the agent randomly selects items first and chooses commands from their Q-functions. EQ uses only local information to calculate each Q-value. We evaluate the models by averaging rewards with 20 independent episodes. The shaded area in each plot indicates 95% confidence intervals in 4 different trials, where all the details of the hyperparameters are provided in [Report, 2019].

5.2 Single Item Selection ($K = 1$)

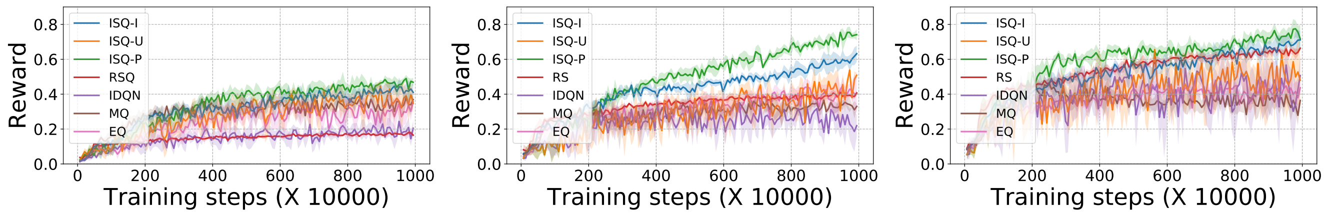
To see the impact of I-sharing, we consider the CS task with $K = 1$, $U = 1$, and $\mathcal{C} = \{\text{stay}\}$, and compare ISQ-I with a vanilla DQN and a sorting DQN. Fig. 4a illustrates the learning performance of the algorithms for $N = 5, 20$, and 50.

Impact of I-sharing The vanilla DQN performs well when $N = 5$, but it fails to learn when $N = 20$ and 50 due to the lack of considering equi-invariance in IS-MDP. Compared to the vanilla DQN, the sorting DQN learns better policies under large N by reducing the state space through sorting. However, ISQ-I still outperforms the sorting DQN when N is large. This result originated from the fact that sorting DQN is affected a lot by the choice of the sorting rule. In contrast, ISQ-I exploits equi-invariance with I-shared Q-network so it can outperform the other baselines for all Ns especially when N is large. The result coincides to our mathematical analysis in Theorem 1 and Theorem 2 which guarantee the expressive power of I-shared Q-network for IS-MDP.



(a) Single selection ($K = 1$) (b) $N = 50, K = 6$ (c) $N = 200, K = 6$

Figure 4: Performances for CS tasks. (a): final performances of the methods for single selection with $N = 5, 20, 50$. (b) and (c): learning curves for $K = 6, U = 0$ with $N = 50, 200$. ISQ-I (1) corresponds to the ISQ-I with a single command 'stay'.



(a) $N = 10, K = 4$ (b) $N = 10, K = 7$ (c) $N = 10, K = 10$

Figure 5: Learning curves for the PP task with 10 predators and 4 preys. Each episode consists of 175 steps.

5.3 Multiple Item Selection ($K > 1$)

To exploit the symmetry in the tasks, we apply I- sharing to all the baselines. For CS task, the environment settings are $K = 6, |\mathcal{C}| = 5$, $U = 0$ and $N = 50, 200$. For PP task, we test with 10 predators ($N = 10$) and 4 preys in a 10×10 grid world for $K = 4, 7, 10$. The learning curves in both CS task (Fig. 4) and PP task (Fig. 5) clearly show that ISQ-I outperforms the other baselines (except other ISQ variants) in most of the scenarios even though we modify all the baselines to apply I-sharing. This demonstrates that ISQ successfully considers the requisites for S-MDP or IS-MDP: a combination of the selected items, command assignment, and future state after the combinatorial selection.

Power of ISQ: Proper Selection Though I-shared Q-networks give the ability to handle large N to all the baselines, ISQs outperform all others in every task. This is because only ISQ can handle all the requisites to compute correct Q-values. IDQN and RSQ perform poorly in many tasks since they do not smartly select the items. RSQ performs much worse than ISQ when $K \ll N$ in both tasks since it only focuses on assigning proper commands but not on selecting good items. Even when $K = N$ (Fig. 5c), ISQ-I is better than RSQ since RSQ needs to explore all combinations of selection, while ISQ-I only needs to explore specific combinations. The other baselines show the importance of future prediction, action selection, and full observation. First, MQ shares the parameters like ISQ-I, but it only considers a reward for the current state. Their difference in performance shows the gap between considering and not considering future prediction in both tasks. In addition, ISQ-I (1) only needs to select items but still has lower performance compared to ISQ-I. This shows that ISQ-I is able to exploit the large action space. Finally, EQ estimates Q-functions using each item's information. The performance gap between EQ and ISQ-I shows the effect of considering full observation in calculating Q-values.

Impact of P-sharing By sharing the parameters in the beginning, ISQ-P learns significantly faster than ISQ-I in all cases as illustrated by the learning curves in Fig. 4 and 5. ISQ-P also outperforms ISQ-U in the PP task because of the increase in the number of parameters at the end of the training process. With these advantages, ISQ-P achieves two goals at once: fast training in early stage and good final performances.

Power of ISQ: Generalization Capability Another advantage of ISQ is powerful generality under environments with different number of items, which is important in real situations. When the number of items changes, a typical Q-network needs to be trained again. However, ISQ has a fixed number of parameters $|\boldsymbol{\theta}|= O(K)$ regardless of N. Therefore, we can re-use the trained θ_k for an item size N_{tr} to re-construct another model for a different item size N_{te}. From the experiments of ISQ-P on different CS scenarios, we observe that for the case $N_{tr} = 50, N_{te} = 200$, ISQ-P shows an 103% performance compared to $N_{tr} = 200, N_{te} = 200$. In contrast, for the case $N_{tr} = 200$ and $N_{te} = 50$, it shows an 86% performance compared to $N_{tr} = 50$ and $N_{te} = 50$. These are remarkable results since the numbers of the items are fourfold different ($N = 50, 200$). We conjecture that ISQ can learn a policy efficiently in an environment with a small number of items and transfer the knowledge to a different and more difficult environment with a large number of items.

6 Conclusion

In this paper, we develop a highly efficient and scalable algorithm to solve continual combinatorial selection by converting the original MDP into an equivalent MDP and leveraging two levels of weight sharing for the neural network. We provide mathematical guarantees for the expressive power of the weight shared neural network. Progressive-sharing share additional weight parameters among K cascaded Q-networks. We demonstrate that our design of progressive sharing outperforms other baselines in various large-scale tasks.

References

[Chen *et al.*, 2018] Xinshi Chen, Shuang Li, Hui Li, Shaohua Jiang, Yuan Qi, and Le Song. Neural model-based reinforcement learning for recommendation. *arXiv preprint arXiv:1812.10613*, 2018.

[Dai *et al.*, 2017] Hanjun Dai, Elias B Khalil, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *NeurIPS*, 2017.

[Dauphin *et al.*, 2014] Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *NeurIPS*, 2014.

[Deudon *et al.*, 2018] Michel Deudon, Pierre Cournut, Alexandre Lacoste, Yossiri Adulyasak, and Louis-Martin Rousseau. Learning heuristics for the tsp by policy gradient. In *CPAIOR*. Springer, 2018.

[Gybenko, 1989] G Gybenko. Approximation by superposition of sigmoidal functions. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

[Jason and Devon R Graham, 2018] Hartford Jason and Siamak Ravanbakhsh Devon R Graham, Kevin Leyton-Brown. Deep models of interactions across sets. In *ICML*, 2018.

[Kawaguchi, 2016] Kenji Kawaguchi. Deep learning without poor local minima. In *NeurIPS*, 2016.

[Kipf and Welling, 2017] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[Kool *et al.*, 2019] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *ICLR*, 2019.

[Kushner and Whiting, 2004] Harold J Kushner and Philip A Whiting. Convergence of proportional-fair sharing algorithms under general conditions. *IEEE Transactions on Wireless Communications*, 3(4):1250–1259, 2004.

[Laurent and Brecht, 2018] Thomas Laurent and James Brecht. Deep linear networks with arbitrary loss: All local minima are global. In *ICML*, 2018.

[Li *et al.*, 2016] Shuai Li, Baoxiang Wang, Shengyu Zhang, and Wei Chen. Contextual combinatorial cascading bandits. In *ICML*, 2016.

[Maes *et al.*, 2009] Francis Maes, Ludovic Denoyer, and Patrick Gallinari. Structured prediction with reinforcement learning. *Machine learning*, 77(2-3):271, 2009.

[Maron *et al.*, 2019] Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the universality of invariant networks. *arXiv preprint arXiv:1901.09342*, 2019.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[Qi *et al.*, 2017] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017.

[Qin *et al.*, 2014] Lijing Qin, Shouyuan Chen, and Xiaoyan Zhu. Contextual combinatorial bandit and its application on diversified online recommendation. In *ICDM*. SIAM, 2014.

[Ravanbakhsh *et al.*, 2017a] Siamak Ravanbakhsh, Jeff Schneider, and Barnabas Poczos. Deep learning with sets and point clouds. In *ICLR, workshop track*, 2017.

[Ravanbakhsh *et al.*, 2017b] Siamak Ravanbakhsh, Jeff Schneider, and Barnabás Póczos. Equivariance through parameter-sharing. In *ICML*, 2017.

[Report, 2019] Report. Solving continual combinatorial selection via deep reinforcement learning. https://github.com/anonybot, 2019.

[Ricci *et al.*, 2015] Francesco Ricci, Lior Rokach, and Bracha Shapira. Recommender systems: Introduction and challenges. In *Recommender systems handbook*. Springer, 2015.

[Tampuu *et al.*, 2017] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 12(4):e0172395, 2017.

[Usunier *et al.*, 2017] Nicolas Usunier, Gabriel Synnaeve, Zeming Lin, and Soumith Chintala. Episodic exploration for deep deterministic policies for starcraft micromanagement. In *ICLR*, 2017.

[Yarotsky, 2018] Dmitry Yarotsky. Universal approximations of invariant maps by neural networks. *arXiv preprint arXiv:1804.10306*, 2018.

[Zaheer *et al.*, 2017] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In *NeurIPS*, 2017.

[Zheng *et al.*, 2018] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. Drn: A deep reinforcement learning framework for news recommendation. In *WWW*, 2018.

[Zinkevich and Balch, 2001] Martin Zinkevich and Tucker Balch. Symmetry in markov decision processes and its implications for single agent and multi agent learning. In *ICML*, 2001.

Appendix A Intra-Parameter Sharing

A.1 Single Channel

In this section, we formally redefine the two types of the previously defined weight shared layers $\phi_k(\cdot)$ and $\psi_k(\cdot)$ with the EI property, i.e., for all $\sigma_s := (\sigma_x, \sigma_i) \in \boldsymbol{S}_k \times \boldsymbol{S}_{N-k}$,

$$\phi_k(\sigma_s(\boldsymbol{x}, \boldsymbol{i})) = \sigma_s(\phi_k(\boldsymbol{x}, \boldsymbol{i})), \qquad \psi_k(\sigma_s(\boldsymbol{x}, \boldsymbol{i})) = \sigma_i(\psi_k(\boldsymbol{x}, \boldsymbol{i})).$$

We start with the simplest case when $\phi_k : \mathbb{R}^{|\boldsymbol{x}|} \times \mathbb{R}^{|\boldsymbol{i}|} \to \mathbb{R}^{|\boldsymbol{x}|} \times \mathbb{R}^{|\boldsymbol{i}|}$ and $\psi_k : \mathbb{R}^{|\boldsymbol{x}|} \times \mathbb{R}^{|\boldsymbol{i}|} \to \mathbb{R}^{|\boldsymbol{i}|}$. This case can be regarded as a state $s = (\boldsymbol{x}, \boldsymbol{i})$ where $\boldsymbol{x} = (x_1, \cdots, x_k) \in \mathbb{R}^k$ and $\boldsymbol{i} = (i_1, \cdots, i_{N-k}) \in \mathbb{R}^{N-k}$ in Section 3.2. Let $\mathbf{I}_x \in \mathbb{R}^{k \times k}$ and $\mathbf{I}_i \in \mathbb{R}^{(N-k)\times(N-k)}$ are the identity matrices. We denote $\mathbf{1}_{x,i} \in \mathbb{R}^{k \times (N-k)}$, $\mathbf{1}_{i,x} \in \mathbb{R}^{(N-k)\times k}$, $\mathbf{1}_{x,1} \in \mathbb{R}^{k \times 1}$, and $\mathbf{1}_{i,1} \in \mathbb{R}^{(N-k)\times 1}$ are the matrices of ones.

Layer ϕ_k Let $\phi_k(\boldsymbol{x}, \boldsymbol{i}) := (\boldsymbol{X}, \boldsymbol{I})$ with $\boldsymbol{x}, \boldsymbol{X} \in \mathbb{R}^k$ and $\boldsymbol{i}, \boldsymbol{I} \in \mathbb{R}^{N-k}$ where the output of the layers $\boldsymbol{X}$ and $\boldsymbol{I}$ are defined as

$$\boldsymbol{X} := \rho(\boldsymbol{W}_x \boldsymbol{x} + \boldsymbol{W}_{x,x} \boldsymbol{x} + \boldsymbol{W}_{x,i} \boldsymbol{i} + \boldsymbol{b}_x), \qquad \boldsymbol{I} := \rho(\boldsymbol{W}_i \boldsymbol{i} + \boldsymbol{W}_{i,i} \boldsymbol{i} + \boldsymbol{W}_{i,x} \boldsymbol{x} + \boldsymbol{b}_i) \tag{6}$$

with a non-linear activation function ρ. The parameter shared matrices $\boldsymbol{W}_x, \cdots, \boldsymbol{W}_{i,i}$ defined as follows:

$$\begin{aligned} &\boldsymbol{W}_x := W_x \mathbf{I}_x, && \boldsymbol{W}_{x,x} := \frac{W_{x,x}}{|\boldsymbol{x}|}\mathbf{1}_{x,x}, && \boldsymbol{W}_{x,i} := \frac{W_{x,i}}{|\boldsymbol{i}|}\mathbf{1}_{x,i}, && \boldsymbol{b}_x := b_x \mathbf{1}_{x,1}, \\ &\boldsymbol{W}_i := W_i \mathbf{I}_i, && \boldsymbol{W}_{i,i} := \frac{W_{i,i}}{|\boldsymbol{i}|}\mathbf{1}_{i,i}, && \boldsymbol{W}_{i,x} := \frac{W_{i,x}}{|\boldsymbol{x}|}\mathbf{1}_{i,x}, && \boldsymbol{b}_i := b_i \mathbf{1}_{i,1}. \end{aligned}$$

The entries in the weight matrices $\boldsymbol{W}_x, \cdots, \boldsymbol{b}_i$ are tied by real-value parameters $W_x, \cdots, b_i \in \mathbb{R}$, respectively. Some weight matrices such as $\boldsymbol{W}_{x,x}, \boldsymbol{W}_{x,i}, \boldsymbol{W}_{i,x}, \boldsymbol{W}_{i,i}$ have normalizing term $1/|\boldsymbol{x}|$ $(= 1/k)$ or $1/|\boldsymbol{i}|$ $(= 1/(N-k))$. In our empirical simulation results, these normalizations help the stable training as well as increase the generalization capability of the Q-networks.

Layer ψ_k The only difference of ψ_k from ϕ_k is that the range of $\psi_k(\boldsymbol{x}, \boldsymbol{i})$ is restricted in $\boldsymbol{I}$ of (6), i.e., $\psi_k(\boldsymbol{x}, \boldsymbol{i}) := \boldsymbol{I} \in \mathbb{R}^{N-k}$ where $\boldsymbol{I} = \rho(\boldsymbol{W}_x \boldsymbol{x} + \boldsymbol{W}_{x,x} \boldsymbol{x} + \boldsymbol{W}_{x,i} \boldsymbol{i} + \boldsymbol{b}_i)$. The weight matrices are similarly defined as in the ϕ_k case:

$$\boldsymbol{W}_i := W_i \mathbf{I}_i, \quad \boldsymbol{W}_{i,i} := \frac{W_{i,i}}{|\boldsymbol{i}|}\mathbf{1}_{i,i}, \quad \boldsymbol{W}_{i,x} := \frac{W_{i,x}}{|\boldsymbol{x}|}\mathbf{1}_{i,x}, \quad \boldsymbol{b}_i := b_i \mathbf{1}_{i,1}.$$

Deep Neural Network with Stacked Layers Recall that the I-shared network $Q_\theta(\cdot\,;\theta_k)$ is formed as follows:

$$Q_\theta(\cdot\,;\theta_k) := \psi_k \circ \phi_k^D \circ \cdots \phi_k^1(\cdot)$$

where D denotes the number of the stacked mutiple layers belonging to ϕ_k. Therefore, the weight parameter vector θ_k for $Q_\theta(\cdot\,;\theta_k)$ consists of $\{W_x^d, \cdots, b_i^d\}_{d=1}^{d=D}$ for ϕ_k and $\{W_i, W_{i,i}, W_{i,x}, b_i\}$ for ψ_k. In contrast, the projected vector $\omega(\theta_k)$ consists of high dimenional weight parameter vectors such as $\{\boldsymbol{W}_x^d, \cdots, \boldsymbol{b}_i^d\}_{d=1}^{d=D}$ for ϕ_k and $\{\boldsymbol{W}_i, \boldsymbol{W}_{i,i}, \boldsymbol{W}_{i,x}, \boldsymbol{b}_i\}$ for ψ_k.

A.2 Multiple Channels

Multiple Channels. In the above section, we describe simplified versions of the intra-sharing layers

$$\phi_k : \mathbb{R}^{|\boldsymbol{x}|} \times \mathbb{R}^{|\boldsymbol{i}|} \to \mathbb{R}^{|\boldsymbol{x}|} \times \mathbb{R}^{|\boldsymbol{i}|}, \qquad \psi_k : \mathbb{R}^{|\boldsymbol{x}|} \times \mathbb{R}^{|\boldsymbol{i}|} \to \mathbb{R}^{|\boldsymbol{i}|}.$$

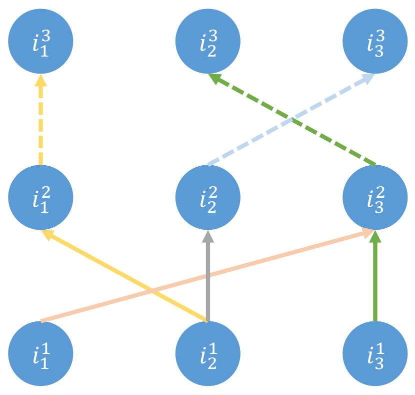
In this section, we extend this to

$$\phi_k : \mathbb{R}^{|\boldsymbol{x}|\cdot P_x + |\boldsymbol{i}|\cdot P_i} \to \mathbb{R}^{|\boldsymbol{x}|\cdot O_x + |\boldsymbol{i}|\cdot O_i}, \qquad \psi_k : \mathbb{R}^{|\boldsymbol{x}|\cdot P_x + |\boldsymbol{i}|\cdot P_i} \to \mathbb{R}^{|\boldsymbol{i}|\cdot O_i} \tag{7}$$

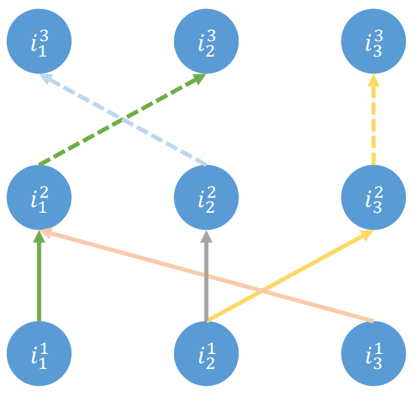
where P_x, P_i, O_x, O_i are the numbers of the features for the input $\boldsymbol{x}, \boldsymbol{i}$ and the output X, I of each layer, respectively. The role of the numbers is similar to that of channels in convolutional neural networks which increase the expressive power and handle the multiple feature vectors. This wideness allows more expressive power due to the increased numbers of the hidden nodes, according to the *universial approximatin theorem* [Gybenko, 1989]. Furthermore, our Theorem 2 also holds with proper feature numbers in the hidden layers. Without loss of generality, we handle the case for $P_x = P_y = P$ and $O_x = O_y = O$. We use superscripts $\boldsymbol{x}^{\langle p \rangle}, \boldsymbol{y}^{\langle p \rangle}$ and $\boldsymbol{X}^{\langle o \rangle}, \boldsymbol{Y}^{\langle o \rangle}$ for $p \in \{1, \cdots, P\}$ and $o \in \{1, \cdots, O\}$ to denote such channels. Our architecture satisfies that cross-channel interactions are fully connected. Layer $\phi_k(\boldsymbol{x}, \boldsymbol{i})$ with multiple channels is as follows:

$$\boldsymbol{X}^{\langle o \rangle} := \rho\left(\sum_{p=1}^{P}\left(\boldsymbol{W}_x^{\langle o,p \rangle}\boldsymbol{x}^{\langle p \rangle} + \boldsymbol{W}_{x,x}^{\langle o,p \rangle}\boldsymbol{x}^{\langle p \rangle} + \boldsymbol{W}_{x,i}^{\langle o,p \rangle}\boldsymbol{i}^{\langle p \rangle} + \boldsymbol{b}_x^{\langle o \rangle}\right)\right),$$

$$\boldsymbol{I}^{\langle o \rangle} := \rho\left(\sum_{p=1}^{P}\left(\boldsymbol{W}_i^{\langle o,p \rangle}\boldsymbol{i}^{\langle p \rangle} + \boldsymbol{W}_{i,i}^{\langle o,p \rangle}\boldsymbol{i}^{\langle p \rangle} + \boldsymbol{W}_{i,x}^{\langle o,p \rangle}\boldsymbol{x}^{\langle p \rangle} + \boldsymbol{b}_i^{\langle o \rangle}\right)\right)$$



(a) Q_ω



(b) $Q_{\sigma(\omega)}$

Figure 6: The example networks with permuted weight parameter vectors by $\sigma \in \boldsymbol{S}_3$ where $\sigma(1) = 3, \sigma(2) = 2, \sigma(3) = 1$. If weights in the different network have the same color then they also share the same weight values.

where

$$\boldsymbol{W}_x^{\langle o,p\rangle} := W_x^{\langle o,p\rangle}\mathbf{I}_x, \qquad \boldsymbol{W}_{x,x}^{\langle o,p\rangle} := \frac{W_{x,x}^{\langle o,p\rangle}}{|\boldsymbol{x}|}\mathbf{1}_{x,x}, \qquad \boldsymbol{W}_{x,i}^{\langle o,p\rangle} := \frac{W_{x,i}^{\langle o,p\rangle}}{|\boldsymbol{i}|}\mathbf{1}_{x,i}, \qquad \boldsymbol{b}_x^{\langle o\rangle} := b_x^{\langle o\rangle}\mathbf{1}_{x,1},$$

$$\boldsymbol{W}_i^{\langle o,p\rangle} := W_i^{\langle o,p\rangle}\mathbf{I}_i, \qquad \boldsymbol{W}_{i,i}^{\langle o,p\rangle} := \frac{W_{i,i}^{\langle o,p\rangle}}{|\boldsymbol{i}|}\mathbf{1}_{i,i}, \qquad \boldsymbol{W}_{i,x}^{\langle o,p\rangle} := \frac{W_{i,x}^{\langle o,p\rangle}}{|\boldsymbol{x}|}\mathbf{1}_{i,x}, \qquad \boldsymbol{b}_i^{\langle o\rangle} := b_i^{\langle o\rangle}\mathbf{1}_{i,1}.$$

Similar to the above cases, the entries in the weight matrices $\boldsymbol{W}_x^{\langle o,p\rangle}, \cdots, \boldsymbol{b}_i^{\langle o\rangle}$ are tied together by real-value parameters $W_x^{\langle o,p\rangle}, \cdots, b_i^{\langle o\rangle}$ respectively. The weight parameter vector θ_k for $Q_\theta(\cdot;\theta_k)$ with multiple channels consists of $\{W_x^d, \cdots, b_i^d\}_{d=1}^{d=D}$ for ϕ_k and $\{W_i, W_{i,i}, W_{i,x}, b_i\}$ for ψ_k. In contrast, the projected vector $\omega(\theta_k)$ consists of high dimenional weight parameter vectors such as $\{\boldsymbol{W}_x^d, \cdots, \boldsymbol{b}_i^d\}_{d=1}^{d=D}$ for ϕ_k and $\{\boldsymbol{W}_i, \boldsymbol{W}_{i,i}, \boldsymbol{W}_{i,x}, \boldsymbol{b}_i\}$ for ψ_k.

Appendix B Proofs of the theorems

B.1 Relative Local optimality: Theorem 1

To simplify the explanation, we only consider the case when phase $k = 0$ so $s = (\boldsymbol{i}) = (i_1, \cdots, i_N)$ and $Q^\star$ is permutation equivariant to the order of $\boldsymbol{i}$. Furthermore, we consider the case of a single channel described in Section A.1. Therefore, we omit to notate k in this subsection and denote σ rather than $\sigma_i \in \boldsymbol{S}_N$. However, our idea for the proof can be easily adapted to extended cases such as $k > 0$ or multiple channels. To summarize, our goal is to show relative local optimality in Theorem 1 where the loss function L_Ω is defined as

$$L_\Omega(\omega) := \sum_{\sigma \in \boldsymbol{S}_N} \sum_{\boldsymbol{i} \in B} \Big| Q_\omega(\sigma(\boldsymbol{i})) - Q^\star(\sigma(\boldsymbol{i})) \Big|.$$

Skectch of Proof To use contradiction, we assume that there exists at least one local minima $\theta^\star \in \Theta$ in the loss function L_Θ for I-shared network Q_θ while $\omega(\theta^\star) \in \Omega$ is not a local minima in the loss function L_Ω for non-weight shared network Q_ω. Therefore, there must be a vector $\omega_0 \in \Omega$ in Ω which makes the directional derivative $D_{\omega_0} L_\Omega(\omega(\theta^\star)) < 0$. We first extend the definition of each $\sigma \in \boldsymbol{S}_N$ to the corresponding mapping $\sigma : \Omega \to \Omega$. We can generate $N!$ more derivative vector $\sigma(\omega_0)$ for each σ such that $D_{\sigma(\omega_0)} L_\Omega(\omega(\theta^\star)) = D_{\omega_0} L_\Omega(\omega(\theta^\star)) < 0$. Therefore, the sum of the whole permuted vectors $\overline{\omega} := \sum_{\sigma \in \boldsymbol{S}_N} \sigma(\omega_0)$ is also a negative derivative vector while belongs to $\omega(\Theta)$ since $\overline{\omega}$ has the effect of I-sharing from the summation of the all permuted derivative vectors. This fact guarantees the existence of a derivative vector $\overline{\theta} \in \Theta$ such that $\overline{\omega} = \omega(\overline{\theta})$ and $D_{\overline{\theta}} L_\Theta(\theta) < 0$ and contradicts to the aformentioned assumption that $\theta^\star$ is the local optimal minima of L_Θ.

Extended Definition for $\sigma \in \boldsymbol{S}_N$ In this paragraph, we will extend the concept of the permutation $\sigma \in \boldsymbol{S}_N$ from the original definition on the set $\{1, 2, \cdots, N\}$ to the permutation on the weight parameter vector ω in non-shared weight parameter vector space Ω, i.e., $\sigma : \Omega \to \Omega$ to satisfy the below statement,

$$\forall \sigma \in \boldsymbol{S}_N, \ \forall \omega \in \Omega, \ \forall \boldsymbol{i} \in \mathbb{R}^N, \quad \sigma(Q_\omega(\boldsymbol{i})) = Q_{\sigma(\omega)}(\sigma(\boldsymbol{i})). \tag{8}$$

To define the permutation with the property in (8), we shall describe how σ permutes weight parameters in a layer $\phi_\omega : \mathbb{R}^N \to \mathbb{R}^N$ in Q_ω, which can be represented as

$$\phi_\omega(\boldsymbol{i}) = \boldsymbol{W}\boldsymbol{i} + \boldsymbol{b} \tag{9}$$

where $\boldsymbol{W} \in R^{N\times N}$ is a weight matrix and $\boldsymbol{b} \in \mathbb{R}^N$ is a biased vector. In the permuted layer $\phi_{\sigma(\omega)}$, the weight matrix $\boldsymbol{W}$ and $\boldsymbol{b}$ in (9) convert to $M_\sigma \circ \boldsymbol{W} \circ M_\sigma^{-1}$ and $M_\sigma \circ \boldsymbol{b}$, respectively. M_σ is a permutation matrix defined as $M_\sigma := [\boldsymbol{e}_{\sigma(1)}, \cdots, \boldsymbol{e}_{\sigma(N)}]$ where $\boldsymbol{e}_n$ is a standard dimensional basis vector in $\mathbb{R}^N$. With the permuted weights, we can easily see $\sigma(\phi_\omega(\boldsymbol{i})) = \phi_{\sigma(\omega)}(\sigma(\boldsymbol{i}))$ for all σ, ω, and $\boldsymbol{i}$. Therefore, the network $Q_{\sigma(\omega)}$ which is a composite of the layers $\phi_{\sigma(\omega)}$s satisfies (8). Figure 6 describes an example of the permutation on ω.

Note that the projected weight parameter vector $\omega(\theta)$ for an arbitrary $\theta \in \Theta$ is invariant to the permutation $\sigma : \Omega \to \Omega$ since $\omega(\theta)$ satisfies the symmetry among the weights from I-sharing, i.e.,

$$\forall\theta \in \Theta, \forall\sigma \in \boldsymbol{S}_N, \quad \omega(\theta) = \sigma(\omega(\theta)). \tag{10}$$

Lemma 1 (Permutation Invariant Loss Function)**.** *For any weight parameter vectors $\omega \in \Omega$, $\theta \in \Theta$, and $\sigma \in \boldsymbol{S}_N$, the below equation holds.*

$$L_\Omega(\omega(\theta) + \omega) = L_\Omega(\omega(\theta) + \sigma(\omega)). \tag{11}$$

(Proof of Lemma 1). We can derive the result of Lemma 1 from the below statement.

$$\begin{aligned}
L_\Omega(\omega(\theta) + \sigma_0(\omega)) &= \sum_{\sigma\in\boldsymbol{S}_N}\sum_{\boldsymbol{i}\in B}\left|Q_{\omega(\theta)+\sigma_0(\omega)}(\sigma(\boldsymbol{i})) - Q^\star(\sigma(\boldsymbol{i}))\right|^2 \\
&= \sum_{\sigma\in\boldsymbol{S}_N}\sum_{\boldsymbol{i}\in B}\left|Q_{\sigma_0(\omega(\theta)+\omega)}(\sigma_0\circ\sigma_0^{-1}\circ\sigma(\boldsymbol{i})) - Q^\star(\sigma_0\circ\sigma_0^{-1}\circ\sigma(\boldsymbol{i}))\right|^2 && (\because (10)) \\
&= \sum_{\sigma\in\boldsymbol{S}_N}\sum_{\boldsymbol{i}\in B}\left|\sigma_0\Big(Q_{\omega(\theta)+\omega}(\sigma_0^{-1}\circ\sigma(\boldsymbol{i}))\Big) - \sigma_0\Big(Q^\star(\sigma_0^{-1}\circ\sigma(\boldsymbol{i}))\Big)\right|^2 && (\because (5), (8)) \\
&= \sum_{\sigma'\in\boldsymbol{S}_N}\sum_{\boldsymbol{i}\in B}\left|Q_{\omega(\theta)+\omega}(\sigma'(\boldsymbol{i})) - Q^\star(\sigma'(\boldsymbol{i}))\right|^2 && (\sigma' := \sigma_0^{-1}\circ\sigma) \\
&= L_\Omega(\omega(\theta)+\omega).
\end{aligned}$$

□

(Proof of Theorem 1). We use contradiction by assumping that there exists a local minima $\theta^\star \in \Theta$ of L_Θ while $\omega(\theta^\star) \in \Omega$ is not a local minima of L_Ω. Since $\omega(\theta^\star)$ is not local minima of L_Ω, there exists a vector $\omega_0 \in \Omega$ such that the directional derivative of $L_\Omega(\omega(\theta^\star))$ along ω_0 is negative, i.e., $D_{\omega_0}(L_\Omega(\omega(\theta^\star))) < 0$. We can find $N!$ additional vectors which have a negative derivative by permuting the $\omega_0 \in \boldsymbol{S}_N$ and exploiting the result of Lemma 1.

$$\begin{aligned}
D_{\sigma(\omega_0)}(L_\Omega(\omega(\theta^\star))) &= \lim_{h\to 0}\frac{L_\Omega(\omega(\theta^\star)) + h\sigma(\omega_0)) - L_\Omega(\omega(\theta^\star))}{h} \\
&= \lim_{h\to 0}\frac{L_\Omega(\omega(\theta^\star) + h\omega_0) - L_\Omega(\omega(\theta^\star))}{h} && (\because (11)) \\
&= D_{\omega_0}(L_\Omega(\omega(\theta^\star))) < 0.
\end{aligned}$$

The existence of the above limit can be induced from the differentiability of the activation function ρ. Furthermore, the activation function is continuously differentiable, so if we set $\overline{\omega} := \sum_{\sigma\in\boldsymbol{S}_N}\sigma(\omega_0)$,

$$D_{\overline{\omega}}(L_\Omega(\omega(\theta^\star))) = \sum_{\sigma\in\boldsymbol{S}_N} D_{\sigma(\omega_0)}(L_\Omega(\omega(\theta^\star))) < 0.$$

From the symmetricity of $\overline{\omega}$ due to the summation of the $N!$ permuted vectors, there exists a vector $\overline{\theta} \in \Theta$ such that $\overline{\omega} = \omega(\overline{\theta})$. Thus, $D_{\overline{\theta}}(L_\Omega(\omega(\theta^\star))) = D_{\overline{\omega}}(L_\Omega(\omega(\theta^\star))) < 0$ which contradicts to the assumption that $\theta^\star$ is the local minima on the loss function L_Ω. □

B.2 Proof of Theorem 2

Sketch of Proof We denote $\mathcal{X} := \mathcal{I} \times \mathcal{C}$ as the domain of the information of the selected items $\boldsymbol{x}$. Recall I-shared Q-network $Q_\theta(\boldsymbol{x}, \boldsymbol{i}) : \mathcal{X}^k \times \mathcal{I}^{N-k} \to \mathbb{R}^{(N-k)\times C}$ and the optimal Q-function $Q^\star(\boldsymbol{x}, \boldsymbol{i}) : \mathcal{X}^k \times \mathcal{I}^{N-k} \to \mathbb{R}^{(N-k)\times C}$ for each phase k share the same input and output domain. We denote $[Q_\theta(\boldsymbol{x}, \boldsymbol{i})]_j \in \mathbb{R}^C$ and $[Q^\star(\boldsymbol{x}, \boldsymbol{i})]_j \in \mathbb{R}^C$ as the jth row of output of $Q_\theta(\boldsymbol{x}, \boldsymbol{i})$ and $Q^\star(\boldsymbol{x}, \boldsymbol{i})$ respectively for $1 \le j \le N-k$. In other words,

$$Q_\theta(\boldsymbol{x}, \boldsymbol{i}) = \begin{bmatrix} [Q_\theta(\boldsymbol{x}, \boldsymbol{i})]_1 \\ \cdots \\ [Q_\theta(\boldsymbol{x}, \boldsymbol{i})]_{N-k} \end{bmatrix}.$$

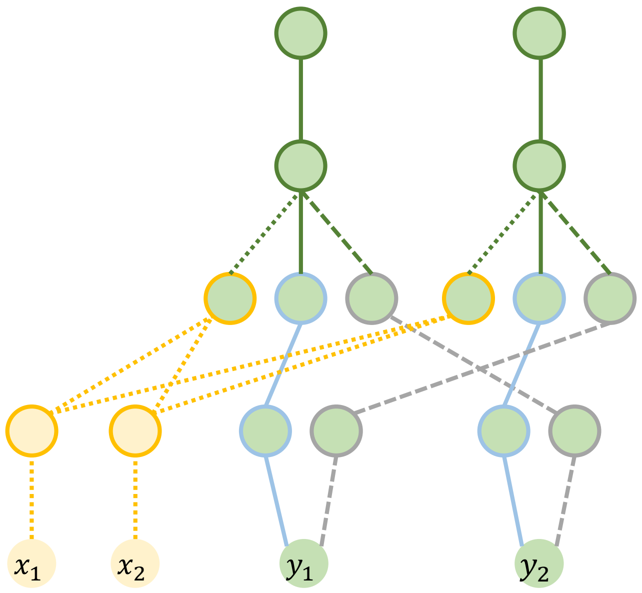


Figure 7: A simplified version of I-shared Q-network $Q_\theta(\boldsymbol{x}, \boldsymbol{i})$ when $N = 4$ and $k = 2$ to approximate $Q^\star(\boldsymbol{x}, \boldsymbol{i}) = H(\sum_{x\in\boldsymbol{x}} \xi_x(x), i_j, \sum_{i\in\boldsymbol{i}_-} \xi_i(i))$. If the edges share the same color and shape in the same layer, the corresponding weight parameters are tied together. The yellow dotted lines represent a mapping to approximate $\xi_x(x)$. The blued solid lines represent an identity mapping. The grey dashed lines represent $\xi_i(i)$. Finally, the green edges generate a mapping to approximate H.

In this proof, we will show that each $[Q^\star(\boldsymbol{x}, \boldsymbol{i})]_j$ can be approximated by $[Q_\theta(\boldsymbol{x}, \boldsymbol{i})]_j$. From the EI property of $Q^\star(\boldsymbol{x}, \boldsymbol{i})$, the jth row $[Q^\star(\boldsymbol{x}, \boldsymbol{i})]_j : \mathcal{X}^k \times \mathcal{I}^{N-k} \to \mathbb{R}^C$ is permutation invariant to the orders of the elements in $\boldsymbol{x}$ and $\boldsymbol{i}_- := (i_1, \cdots, i_{j-1}, i_{j+1}, \cdots, i_{N-k})$ respectively, i.e.,

$$\forall \sigma_x \in \boldsymbol{S}_k,\ \forall \sigma_{i_-} \in \boldsymbol{S}_{N-k-1}, \quad [Q^\star(\boldsymbol{x}, i_j, \boldsymbol{i}_-)]_j \equiv [Q^\star(\sigma_x(\boldsymbol{x}), i_j, \sigma_{i_-}(\boldsymbol{i}_-))]_j. \tag{12}$$

In Lemma 2, we show that $[Q^\star(\boldsymbol{x}, i_j, \boldsymbol{i}_-)]_j$ can be decomposed in the form of $H(\sum_{x\in\boldsymbol{x}} \xi_x(x), i_j, \sum_{i\in\boldsymbol{i}_-} \xi_i(i))$ where H, ξ_x, ξ_y are proper continuous functions. Finally, we prove that I-shared Q-network Q_θ with more than four layers can approximate the decomposed forms of the functions: H, ξ_x,and ξ_y.

Lemma 2. *If a continuous function $F(\boldsymbol{x}, i, \boldsymbol{i}_-) : \mathcal{X}^k \times \mathcal{I} \times \mathcal{I}^{N-k-1} \to \mathbb{R}^C$ is permutation invariant to the orders of the items in $\boldsymbol{x} \in \mathcal{X}^k$ and $\boldsymbol{i}_- \in \mathcal{I}^{N-k-1}$, i.e.,*

$$\forall \sigma_x \in \boldsymbol{S}_k,\ \forall \sigma_{i_-} \in \boldsymbol{S}_{N-k-1}, \quad F(\sigma_x(\boldsymbol{x}), i, \sigma_{i_-}(\boldsymbol{i}_-)) \equiv F(\boldsymbol{x}, i, \boldsymbol{i}_-).$$

if and only if $F(\boldsymbol{x}, i, \boldsymbol{i}_-)$ can be represented by proper continous functions H, ξ_x, and ξ_i with the form of

$$F(\boldsymbol{x}, i, \boldsymbol{i}_-) = H\Big(\sum_{x\in\boldsymbol{x}} \xi_x(x), i, \sum_{i\in\boldsymbol{i}_-} \xi_i(i)\Big). \tag{13}$$

Proof. The sufficiency is easily derived from the fact that $\sum_{x\in\boldsymbol{x}} \xi_x(x)$, and $\sum_{i\in\boldsymbol{i}_-} \xi_i(i)$ are permutation invariant to the orders of $\boldsymbol{x}$ and $\boldsymbol{i}_-$ respectively. Therefore, $H\Big(\sum_{x\in\boldsymbol{x}} \xi_x(x), i, \sum_{i\in\boldsymbol{i}_-} \xi_i(i)\Big)$ must be permutation invariant to the orders of $\boldsymbol{x}$ and $\boldsymbol{i}_-$.

To prove the necessity, we exploit a result of Theorem 7 in [Zaheer *et al.*, 2017] about the existences of following continuous functions with proper compact sets $\mathcal{X}_0$ and $\mathcal{I}_0$ on Euclidean space.

$$\begin{array}{lll} \exists \eta_x : \mathcal{X}_0^{k+1} \to \mathcal{X}^k, & \exists \xi_x : \mathcal{X} \to \mathcal{X}_0^{k+1}, & \eta_x(\sum_{x\in\boldsymbol{x}} \xi_x(x)) := \boldsymbol{x}, \\ \exists \eta_i : \mathcal{I}_0^{N-k} \to \mathcal{I}^{N-k-1}, & \exists \xi_i : \mathcal{I} \to \mathcal{I}_0^{N-k}, & \eta_i(\sum_{i\in\boldsymbol{i}} \xi_i(i)) := \boldsymbol{i}. \end{array} \tag{14}$$

Therefore, we can define a continuous function $H(\cdot,\cdot,\cdot) : \mathcal{X}_0^{k+1} \times \mathcal{I} \times \mathcal{I}_0^{N-k} \to \mathbb{R}^C$ as

$$H(\cdot,\cdot,\cdot) := F(\eta_x(\cdot),\ \cdot, \eta_i(\cdot)).$$

It is obvious that the function H satisfies (13). □

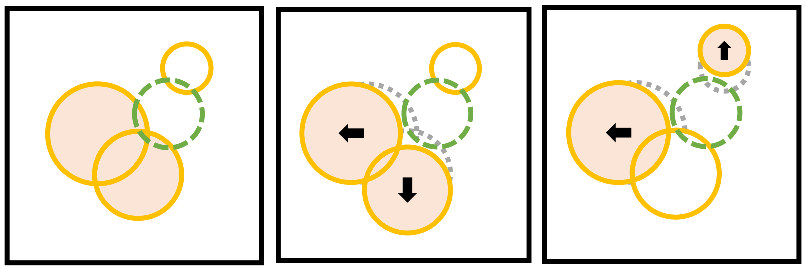

(a) $r < 0$ (b) $r = 0$ (c) $r > 0$

Figure 8: Example scenarios of the CS task with $N = 3$ selectable (orange colored) and $U = 1$ unselectable (green dashed) circles, with $K = 2$ selected (shaded) circles. The assigned commands are represented by the arrows. The agent receives (a) *negative* reward if selected circles overlap with unselectable one; (b) *zero* reward if only selected circles are overlapped with each other; and (c) *positive* reward if there is no overlap.

Proof. With the result of the lemma, the only remained problem to be checked is that I-shared Q-network $Q_\theta(\boldsymbol{x}, i_j, \boldsymbol{i}_-)$ with 4 layers is able to approximate $H(\sum_{x\in\boldsymbol{x}} \xi_x(x), i_j, \sum_{y\in\boldsymbol{i}_-} \xi_y(y))$ if the size of the nodes increases. During this proof, we use the universal approximation theorem by [Gybenko, 1989] which shows that any continuous function f on a compact domain can be approximated by a proper 2-layered neural network. To approximate functions of the decomposition, we can increase the number of the channels described in Section A.2. We omit the biased term $\boldsymbol{b}$ for simplicity. Figure 7 describes the architecture of Q_θ. For ξ_x, there exist weight parameter vectors M and M' in θ such that $\xi_x \approx M \circ M'$. We set $W_x^1 := M'$ and $W_{x,i}^2 := M$ (Apricot edges). Similarly, we can also find weight parameter vectors $W_i^1 := \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & R' \end{bmatrix}$ and $W_{i,i}^2 := R$ where $\xi_i \approx R \circ R'$ (Grey edges). The identity in W_i^1 and $W_i^2 = \boldsymbol{I}$ (Blue edges) represent the passing i_j as the input of H. We set W_i^3 and W_i^4 to satisfy $H \approx W_i^4 \circ W_i^3$ (Green edges). Other weight parameters such as $W_{i,i}^3$ just have zero values. With this weight parameter vector for Q_θ, $Q_\theta(\boldsymbol{x}, i_j, \boldsymbol{i}_{-j})$ successfully approximates the function $H\Big(\sum_{x\in\boldsymbol{x}} \xi_x(x), i, \sum_{i\in\boldsymbol{i}_-} \xi_i(i)\Big) = [Q^\star(\boldsymbol{x}, i, \boldsymbol{i}_-)]_j$ which is jth row values of $Q^\star$. Furthermore, the EI property also implies that for all j, $[Q_\theta(\boldsymbol{x}, i, \boldsymbol{i}_-)]_j$ are the same function, in fact. Therefore, the I-shared Q-network Q_θ with this architecture can approximate all the rows of $Q^\star$ simultaneously. $\square$

Appendix C Detailed Experiment Settings

In this subsection, we explain the environment settings in more detail.

C.1 Evaluation Settings

Circle Selection (CS) As mentioned in Section 5.1, the game consists of N selectable and U unselectable circles within a 1×1 square area, as shown in Fig. 8. Here, circles are the items and $i_n := (pos_x, pos_y, rad)$ are their contexts, where pos_x and pos_y are their center coordinates. Initially, all circles have random coordinates and radius, sampled from $(pos_x, pos_y) \in [-0.5, 0.5] \times [-0.5, 0.5]$ and $rad \in [0, 0.45]$ respectively, After the agent selects K circles with the allocated commands, transition by S-MDP occurs as follows. The selected circles disappear. The unselectable circles that collide with the selected circles disappear. New circles replace the disappeared circles, each of initial radius 0.01 with uniformly random position in $[-0.5, 0.5] \times [-0.5, 0.5]$. Remaining circles expand randomly by $[0.045, 0.055]$ in radius (until maximum radius 0.45) and move with a noise sampled uniformly from $[-0.01, 0.01] \times [-0.01, 0.01]$. The agent also receives reward r after the Kth selection, calculated for each selected circle k of area A_k as follows: *Case 1.* The selected circle collides with one or more unselectable circle: $r = -A_k$. *Case 2.* Not case 1, but the selected circle collides with another selected circle: $r = 0$. *Case 3.* Neither case 1 nor 2: $r = A_k$. We test our algorithm when $K = 6$ with varying $N = 50, 200$. This fact is described in Figure 8.

C.2 Predator-Prey (PP)

In PP, N predators and U preys are moving in $G \times G$ grid world. After the agent selects K predators as well as the commands, the transition in S-MDP occurs. In our experiments, we tested the baselines when $N = 10, U = 4$ with varying $K = 4, 7, 10$ while $G = 10$. A reward is a number of the preys that are caught by more than two predators simultaneously. For each prey, there are at most 8 neighborhood grids where the predator can catch the prey.

C.3 Intra-sharing with unselectable items

In real applications, external context information can be beneficial for the selection in S-MDP. For instance, in the football league example, the enemy team's information can be useful to decide a lineup for the next match. ISQ can handle this contextual information easily with a simple modification of the neural network. Similar to invariant part for previously selected items (red parts in Fig. 2) of I-shared Q-network, we can add another invariant part in the Q-networks for the external context: the information of the unselectable circles (CS) and prey (PP).

C.4 Hyperparameters

During our experiment, we first tuned our hyperparameters for CS and applied all hyperparmeters to other experiments. The below table shows our hyperparameters and our comments for Q-neural networks.

Table 1: Training hyperparameters

Hyperparameter	**Value**	**Descriptions**
Replay buffer size	$50,000$	Larger is stable
Minibatch size	64	Larger performs better
Learning rate (Adam)	0.001	Larger is faster and unstable
Discount factor	0.99	Discount factor γ used in Q-learning update
Target network update frequency	1000	The larger frequency (measured in number of training steps) becomes slower and stable
Initial exploration	1	Initial value of ϵ used in ϵ-greedy exploration
Final exploration	0.1	Final value of ϵ used in ϵ-greedy exploration
Number of layers	3	The number of the layers in the Q-network
Number of nodes	48	The number of channels per each item in a layer
Random seed	4	The number of random seeds for the independent training

C.5 Computation cost

We test all baselines on our servers with Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz (Cpu). Our algorithm (ISQ-I) able to run 1.1610^6 steps during one day in CS ($N = 200$, $K = 6$). Usually, ISQ-I is robust to large N from I-sharing. However, the computation time linearly increases as K grows since the number of the networks should be trained increases large. This problem will be fixed if we exploit the parallelization with GPUs.