Solving Continual Combinatorial Selection via Deep Reinforcement Learning

Hyungseok Song[1*], **Hyeryung Jang**[2], **Hai H. Tran**[1], **Se-eun Yoon**[1], **Kyunghwan Son**[1], **Donggyu Yun**[3], **Hyoju Chung**[3], **Yung Yi**[1]

[1]School of Electrical Engineering, KAIST, Daejeon, South Korea
[2]Informatics, King's College London, London, United Kingdom
[3]Naver Corporation, Seongnam, South Korea

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 the number of items is 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 symmetry in IS-MDPs with novel weight shared Q-networks, which provably maintain sufficient expressive power. 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 to participate in it and allocating one of C positions (*command*: c) to each of them, with possible duplication. You can observe a collection (*state*: $\tilde{s}$) of the current status (*information*: i_n) of each candidate player (*item:* n). 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 which are stochastically determined by a transition probability function $\tilde{\mathcal{P}}$. 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 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 selecting all at once. IS-MDP significantly relieves the complexity of the joint action space per state in S-MDP; the agent only needs to evaluate KNC actions during K consecutive steps in IS-MDP, while it considers $\binom{N}{K}C^K$ 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 levels of weight parameter sharing for Q-networks: intra-sharing (I-Sharing) and unified-sharing (U-Sharing). In pactice, we propose to use a mixture of I- and U-sharing, which we call progressive sharing (P-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

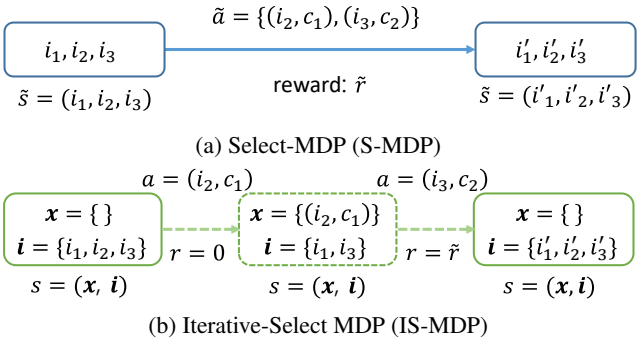
*Contact Author: 7590sok@gmail.com

theorectically claim that I-sharing does not hurt the expressive power for IS-MDP 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 [Qi *et al.*, 2017]. 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] 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.



(a) Select-MDP (S-MDP)

(b) Iterative-Select MDP (IS-MDP)

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

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 $\mathcal{A}(s) := \{(n, c) \,|\, n \in \{1, \dots, N - k\}, c \in \mathcal{C}\}$, 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,

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

Recall $\tilde{\mathcal{P}}$ is the transition probability of S-MDP. The decomposition of joint action in S-MDPs (i.e., selecting K items at once) into K consecutive selections in IS-MDPs has equivalence in terms of the optimal policy [Maes *et al.*, 2009]. Important advantage from the decomposition is that IS-MDPs

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

have action space $\mathcal{A}$ of size NC while the action space of S-MDPs is $\binom{N}{K}C^K$.

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, i.e., for all $s, s' \in \mathcal{S}$, $a \in \mathcal{A}$, and $r \in \mathcal{R}$,

$$\mathcal{P}(s', r \mid s, a) \equiv \mathcal{P}(s', r \mid \sigma_s(s), \sigma_i(a)). \quad (3)$$

We denote $\sigma_s = (\sigma_x, \sigma_i) \in \boldsymbol{S}_k \times \boldsymbol{S}_{N-k}$ as a permutation of a state s at phase k, which is defined as

$$\sigma_s(s) := (\sigma_x(\boldsymbol{x}), \sigma_i(\boldsymbol{i})), \quad (4)$$

where $\boldsymbol{S}_k$ is a group of permutations of a set with k elements. From (3), we can easily induce that if the action $a = (n, c) \in \mathcal{A}(s)$ is the optimal action for s, then for state $\sigma_s(s)$, an optimal policy should know that an permuted action $\sigma_i(a) := (\sigma_i(n), c)$ is also optimal. As a result, we have $\forall s \in \mathcal{S}, \forall a \in \mathcal{A}(s)$,

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

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 (5).

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 $\sigma_s = (\sigma_x, \sigma_i)$,*

$$Q^\star(\sigma_s(s)) = \sigma_i(Q^\star(s)). \quad (6)$$

As we will discuss later, this EI property in (6) 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 $\mathcal{A}(s) = \{(n, c) \mid n \in \{1, \ldots, N-k\}, c \in \mathcal{C}\}$. 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 (6), 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.

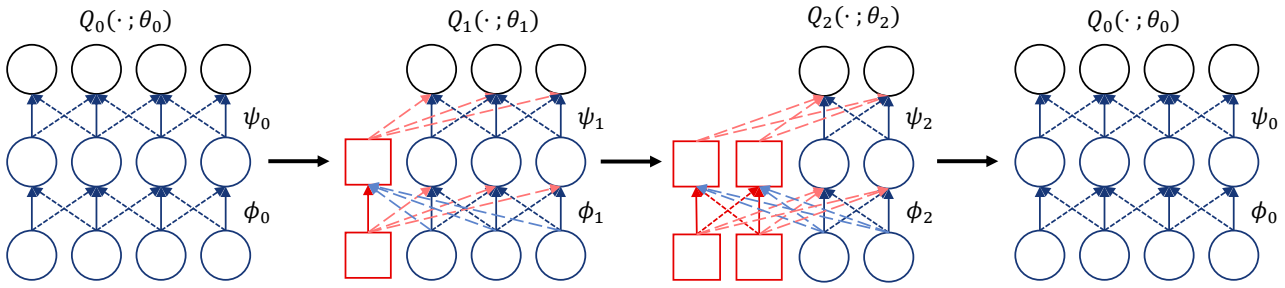


Figure 2: A simple example of the parameter-shared Q-networks $Q_k(\cdot\,;\theta_k)$ when $K = 3, N = 4, |\mathcal{C}| = 1$. Red and blue colored nodes represent the nodes equivariant to the selected items $\boldsymbol{x}$ and the unselected items $\boldsymbol{i}$ respectively. Each black node represents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

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, an item which must be selected during K phases has the same Q-values in every phase.[4] 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

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.

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

4 Intra-Sharing: Relativ Local 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 $Q_k^\star(s) : \mathcal{S}_k \to \mathbb{R}^{|\mathcal{A}(s)|}$ 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 $Q_k^\star(s)$ with the EI property in (6). 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. 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 $Q_k^\star(s)$ as $Q^\star(s)$, $Q_k(s\,;\omega_k)$ as $Q_\omega(s)$, and $Q_k(s\,;\theta_k)$ as $Q_\theta(s)$.

[3] https://github.com/selectmdp

[4] Note that, we set the discount factor $\gamma = 0$ during except the final phase $K - 1$.

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.

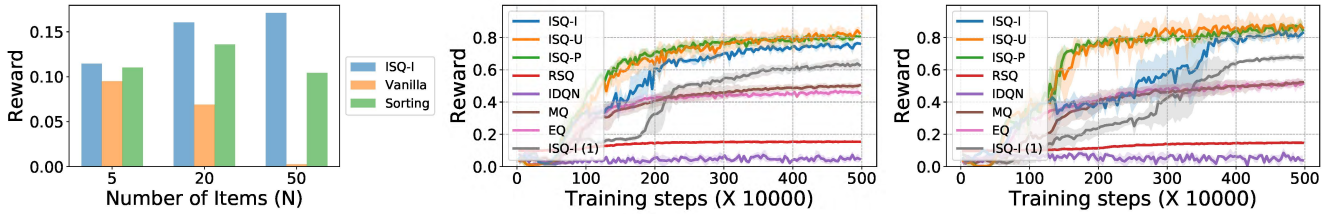
Tested Algorithms and Setup We compare the three variants of ISQ: ISQ-I, ISQ-U, ISQ-P with three DQN-based schemes: (i) a vanilla DQN [Mnih *et al.*, 2015], (ii) a sorting DQN that reduces the state space by sorting the order of items based on a pre-defined rule, and (iii) a myopic DQN which learns to 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 ourthe technical report[5].

5.2 Single Item Selection

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.

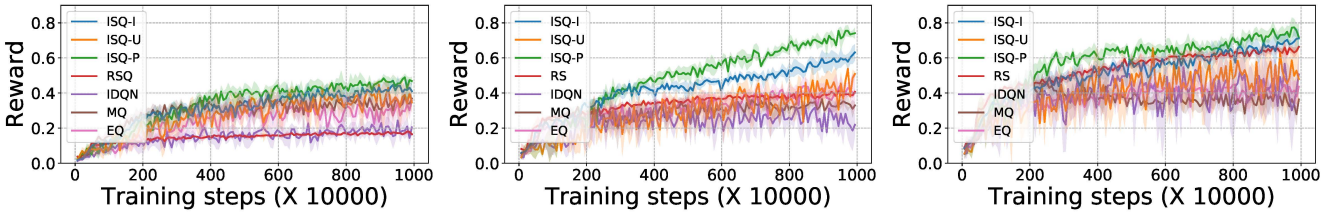
Impact of I-sharing The vanilla DQN performs well when $N = 5$, but it fails to learn when $N = 20$ and 50 due to the lack of considering equi-invariance in IS-MDP. Compared to the vanilla DQN, the sorting DQN learns better policies under large N by reducing the state space through sorting.

[5] https://github.com/selectmdp



(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.

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.

5.3 Multiple Item Selection

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 reuse 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.

Acknowledgements

This work was supported by Institute for Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No.2016-0-00160, Versatile Network System Architecture for Multi-dimensional Diversity).

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science and ICT (No. 2016R1A2A2A05921755).

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.

[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.

[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.