An Approximate Dynamic Programming Algorithm for Multi-Stage Capacity Investment Problems

Sixiang Zhao
Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore 117576, zhaosixiang@u.nus.edu

William B. Haskell
Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore 117576, isehwb@nus.edu.sg

Michel-Alexandre Cardin
Dyson School of Design Engineering, Imperial College London, London SW7 2AZ, United Kingdom, m.cardin@imperial.ac.uk

This paper studies a dynamic multi-facility capacity investment problem (MCIP) with discrete capacity. In this setting, capacity adjustment decisions are made sequentially based on observations of demand. First, we formulate this problem as a Markov decision process (MDP). Then, we design a customized fitted value iteration (FVI) algorithm. In particular, we approximate the value functions with a two-layer neural network with piecewise linear activation functions. However, the action selection procedure of FVI for MCIP can be time-consuming since the action space is discrete and high dimensional. To speed up the action selection, we recast the action selection problem as a two-stage stochastic programming problem. The resulting recourse function comes from the two-layer neural network, and it is solved with a specialized multi-cut decomposition algorithm. Numerical studies show that our algorithm provides high quality solutions to the MCIP, and also that the multi-cut algorithm can significantly speed up the action selection problem of FVI in comparison to the brute-force method and the integer L-shape algorithm. Finally, we show that the techniques developed here for MCIP are applicable to many other finite-time horizon MDPs with finite but high dimensional action spaces.

Key words: real options analysis, multi-facility capacity investment problem, discrete capacity, approximate dynamic programming, decomposition algorithm

1. Introduction

Strategic capacity decisions are important to companies because of the high expenditures entailed and the uncertainty associated with the business environment. To deal with the uncertainty, a wiser decision is to adjust the capacity periodically given the new information of the uncertain
parameters, such as demands, instead of establishing facilities with a large amount of capacity in the beginning. Real options analysis has provided an efficient framework to evaluate the value of a system with dynamically adjusted capacity (Trigeorgis 1996). It views capacity investment decisions as a series of options, i.e. capacity adjustment options, that can be dynamically exercised. Namely, in each decision epoch, decision makers have the right, but not the obligation, to invest or salvage the capacity once new demand information is revealed. A key question in real options analysis is to evaluate the economic performance of the system with capacity adjustment options over a system without such options (i.e., the value of flexibility). Evaluating the performance of a system with such options is a multi-stage decision making problem under uncertainty, of which the optimal capacity investment policy is a function with respect to the observed uncertain parameters. In a capacity investment problem with a single facility, the capacity investment policy is to determine when and how much to adjust the capacity when the uncertain customer demands are observed over time. In a multi-facility problem, the system has not only the options to adjust capacity, but also the options to switch between facilities; for example, if one facility runs out of capacity, the excess demands can be transported to adjacent facilities by paying extra fees. In this context, evaluating the economic performance becomes harder as the decisions are multi-dimensional. Though multi-dimensional, the form of the optimal policy can still be found when the capacity investment problem is convex—i.e. the capacity is continuous and the capacity adjustment costs are convex (Eberly and Van Mieghem 1997). However, these assumptions are too strict in practice as the capacity of a production system is usually modular-designed; that is, the capacity is discrete.

Motivated by the issues outlined above, we study a multi-facility capacity investment problem (MCIP) with discrete capacity in this paper. Such problems have been widely studied in many fields, such as transportation systems (Sun and Schonfeld 2015), the semiconductor industry (Huang and Ahmed 2009), and energy systems (Singh et al. 2009). To solve this problem, one prevalent method is multi-stage stochastic programming, where the evolution of the uncertain
parameters is modeled as a scenario tree and the problem solved as a large-scale mixed integer programming problem. However, the scenario tree grows exponentially with the number of stages and the dimension of the uncertain parameters, so this method can easily become intractable. Another method that can solve MCIP is the decision rule-based multi-stage stochastic programming, where the policy space of the problem is approximated given a preset functional space. Zhao et al. (2018) have designed an if-then decision rule to solve a multi-facility capacity expansion problem with uncertain demands, but the proposed decision rule may require redesigns once capacity contraction or other types of uncertainty, such as price, are considered.

As an alternative, the MCIP can be formulated as a Markov decision process (MDP), and solved via dynamic programming (DP)-based algorithms. However, DP-based methods are subject to the curse of dimensionality; that is to say, the complexity of the algorithm increases exponentially with the number of customers and of facilities. To alleviate the burden brought by the high dimensional state space, we can use approximate dynamic programming (ADP)—more specifically, we can use the fitted value iteration (FVI) algorithm—to approximate the value functions of the problem. In FVI, the value function of the last stage is firstly approximated and then the algorithm proceeds via going backward in time. In each time period, given each generated sample from the state space, one needs to choose the optimal action by solving an action selection problem. Since the action space is finite, the selection problem can be done by enumeration, but this method can be extensively time-consuming in MCIP, whose action space is multi-dimensional.

In this paper, we propose a neural network-based fitted value iteration (NN-FVI) to solve MCIPs, where the capacity can be either expanded or contracted. Then, a decomposition algorithm is proposed to accelerate the action selection procedure of NN-FVI. Theoretical guarantee of the algorithm is provided, and the numerical studies herein show that it can speed up NN-FVI significantly in comparison with the enumeration method. More specifically, the contributions of this paper are summarized below:

1. We analyze the functional form of the value functions of MCIP, and show that the value functions present some piecewise linear structure. Then, we approximate the value functions by
two-layer neural networks with rectified linear units (ReLU) being activation functions. We also prove that the NN-FVI algorithm with ReLU is consistent; that is, an $\epsilon$-optimal solution with probability $\delta$ can be achieved under some mild conditions.

2. Since the action space of MCIP is multi-dimensional, it can be time-consuming in selecting the optimal action via enumeration. To accelerate the action selection procedure, we formulate it as a two stage stochastic programming problem with a non-convex recourse function where the recourse function is derived from the two-layered neural network. This type of stochastic programming problems has been widely studied in (Laporte and Louveaux 1993, Sen and Sherali 2006, Sen et al. 2010), but our method differs from the literature in that we design a multi-cuts method by making use of the gradient information of the recourse function. We also show that the proposed multi-cut algorithm can converge to the global optimum in a finite number of steps, and the numerical studies herein show that it can speed up the algorithm significantly in comparison with the enumeration method or the integer L-shape algorithm.

3. We explain how to extend our method to other dynamic programming problems, such as MCIP with lead time or MCIP with uncertain rewards/costs; the proposed algorithm works as long as problems have combinatorial action selection and the value functions are approximated via two-layer neural networks with ReLU activation functions.

The remainder of the paper is organized as below. Related work about multi-stage capacity planning problems and their solutions are reviewed in Section 2. In Section 3, an MCIP model with discrete capacity is presented. The proposed NN-FVI, along with the analysis of the value functions, are presented in Section 4. Section 5 introduces the proposed multi-cut decomposition algorithm to solve the action selection procedure. The numerical studies are presented in Section 6 and we discuss the extensions of the proposed method in Section 7. In the last section, we conclude the major findings and arguments of this study.
2. Related Work

2.1. Capacity Investment Problems

Since the seminal work (Manne 1961), capacity investment problems with stochastic demands have been widely studied. Analytical results for capacity expansion problem with a single facility and discrete capacity have been derived by (Angelus et al. 2000), where the authors found that the optimal policy of the problem is similar to the well-known \((s,S)\) policy in inventory management. More recently, papers investigated problems where contraction of capacity is allowed (Angelus and Porteus 2002). Ye and Duenyas (2007) studied a capacity investment problem with two-sided fixed costs, and characterized the structure of the optimal policy. This result was further extended by considering risk aversion of decision makers (Lu and Yan 2016). For multi-facility problems, Eberly and Van Mieghem (1997) studied a multi-factors capacity investment problem and derived the structure of the optimal policy, but the result is subject to the assumptions that the capacity is continuous and the capacity adjustment costs are convex. Two-stage multi-facility capacity expansion models with discrete capacity have been studied in the semiconductor industry (Geng et al. 2009) or in transportation systems (Dong et al. 2015), where the first stage sets up capacity plan and the second stage includes the wait-and-see allocation decisions among the installed facilities. However, the two-stage models are inflexible as the capacity investment plan is unchanged during the planning horizon, regardless of the realizations of uncertainty. Huang and Ahmed (2009) studied a multi-facility capacity expansion problem with discrete capacity, and derived an analytical bound for the value of the multi-stage model over that of the two-stage model. Truong and Roundy (2011) developed a multidimensional approximation algorithm to solve capacity expansion problems by decomposing the problem via a cost-separation scheme. Similar applications about multi-stage multi-facility capacity expansion models with discrete capacity can also be found in waste-to-energy systems (Zhao et al. 2018), or in mobility on-demand transportation (Cardin et al. 2017a). For comprehensive review, interesting readers can also refer (Luss 1982, Van Mieghem 2003, Martínez-Costa et al. 2014) for details.
2.2. Stochastic Programming Methods

One prevalent method to solve multi-period optimization problems under uncertainty is multi-stage stochastic programming. Ahmed et al. (2003) solved a multi-facility capacity expansion problem via a scenario tree-based multi-stage stochastic programming. Huang and Ahmed (2009) and Taghavi and Huang (2016) decomposed the scenario tree-based model across facilities, and then solved the sub-problems via a linear programming (LP) based approximation algorithm. Taghavi and Huang (2018) considered a similar problem with budget constraints, and solved it with a heuristic Lagrangian relaxation method. Another algorithm that can solve the MCIP with discrete capacity is stochastic dual dynamic integer programming (Zou et al. 2018); it formulates the uncertain parameters as a scenario tree and the problem is solved based on a nested decomposition scheme. However, when solving MCIP, the number of nodes of a scenario tree not only increases exponentially with the number of stages but also with the dimensions of the uncertain parameters, which make the size of the tree astronomical and the convergence of the algorithm slow.

Decision rule-based methods are proposed to solve multi-stage capacity planning problems. A decision rule is a function which maps the realized uncertainty to the decisions; it approximates the policy space of a multi-stage problem by specifying the functional form of the policy (Shapiro et al. 2009). In the previous studies, if-then decision rules have been proposed to solve a multi-stage capacity expansion problem. An if-then decision can be stated as: if the gap of capacity exceeds a certain threshold, then expand the capacity to a certain level (Cardin et al. 2017b). Zhao et al. (2018) extended the if-then decision rule to a multi-facility problem by introducing a weighted matrix, and designed a decomposition algorithm to optimize the control parameters of the rule. Unfortunately, these methods only work for systems with capacity expansion options only. If one needs to extend the if-then decision rules in solving the problems with capacity contraction, the proposed rules require redesigns: another if-then statement for capacity contraction is needed, and it may complicate the problem because additional binary variables are introduced. In addition, the proposed if-then rules do not work if other uncertain parameters, besides demands, are introduced.
2.3. Dynamic Programming Methods

As another alternative, approximate dynamic programming (ADP), or reinforcement learning, incorporates the capacity decisions and uncertain parameters into the underlying MDP. The basic idea of ADP is to approximate the value functions or $Q$ functions via state aggregation (Bertsekas 2012), basic functions approximation (Munos and Szepesvári 2008, Antos et al. 2008, Powell 2011), or sample average approximation (Haskell et al. 2016). Many of the ADPs in the literature assumed that the action space of the MDP is finite, so that we can enumerate all possible actions in the action selection procedure. However, it can be extensively time-consuming to find the optimal action via enumeration when the action space is multi-dimensional. Policy gradient algorithms have been proposed to find the optimal action for problems with continuous and high-dimensional action spaces (Lillicrap et al. 2015, Schulman et al. 2015), but this type of methods may not work for derivative-free problems such as MCIPs with discrete capacity. Other methods to handle with the derivative-free problems include direct policy search—the policy of the problem is approximated, for example, via neural networks—and the optimization of such policy is done by a black-box optimization (Hu et al. 2017). Jain and Varaiya (2010) approximated the policies via sampling and proposed a simulation-based optimization framework for policy improvements. However, the required number of samples of this type of methods could be large. The NN-FVI algorithm studied in this paper has been partly presented in a conference paper (Zhao et al. 2017), but Zhao et al. (2017) studied a problem with capacity expansion only and the proposed algorithm selects the optimal action via enumeration.

3. Problem Description

**Notation.** Throughout this paper, we denote $0_N$ as an $N$-dimensional vector of zero. Denote $\mathbb{Z}$ as the set of integers and $\mathbb{Z}_+$ as the set of non-negative integers. We further denote $\text{supp}(\cdot)$ as the support of a random variable. Let $\text{dim}(\cdot)$ and $\text{ext}(\cdot)$ denote the dimensions and extreme points of a set, respectively. We denote $\|\cdot\|_p$ as the $p$-norm for vectors; if $p = 2$, we use $\|\cdot\|$ instead of $\|\cdot\|_2$. Finally, for a real-valued measurable function $f$ and a probability distribution $\mu$ defined over $X$, we define the $\mathcal{L}^p$-norm by $\|f\|_{p,\mu} \triangleq (\int_X |f(x)|^p \mu(dx))^{1/p}$. 
Consider a multi-stage stochastic capacity investment problem with customers $\mathcal{I} \triangleq \{1, \ldots, I\}$ and facilities $\mathcal{N} \triangleq \{1, \ldots, N\}$. In each time period, customer demand is allocated to and satisfied by the facilities. The objective is to maximize the net present value (NPV) by determining the optimal capacity investment plan and demand allocation plan over the finite planning horizon $\mathcal{T} \triangleq \{1, \ldots, T\}$. Denote $D_{it}$ as the random demand from customer $i \in \mathcal{I}$ in time period $t \in \mathcal{T}$, and denote its vector form as $D_t = (D_{1t}, \ldots, D_{It})$. Denote $d_{it}$ as a realization of $D_{it}$, with vector form $d_t = (d_{1t}, \ldots, d_{It})$. Let $K_t = (K_{1t}, \ldots, K_{Nt})$ be the installed capacity vector at the end of time period $t$, and let $K_0 = (K_{10}, \ldots, K_{N0})$ be the initial capacity vector. Define $\mathbb{D}$ to be the domain of the demand, i.e. $\mathbb{D} \triangleq \bigcup_{t \in \mathcal{T}} \text{supp}(D_t)$. Define $\mathbb{K}$ as the domain of the capacity and $K_{\text{max}} = (K_{1\text{max}}, \ldots, K_{N\text{max}})$ as a vector of capacity limits (we assume the capacity is bounded), so that

$$\mathbb{K} \triangleq \{K \in \mathbb{Z}^N_+ | 0 \leq K \leq K_{\text{max}}\}.$$  

Our main assumptions are listed below.

**Assumption 1.** The process, $\{D_t, t \in \mathcal{T}\}$, is a Markov process with $I$ components. Without loss of generality, we assume $D_t$ is independent of the installed capacity $K_{t-1}$ for all $t \in \mathcal{T}$.

Given Assumption 1, if denote $P(\cdot | d_{t-1})$ to be the conditional probability function, then we have $D_t \sim P(\cdot | d_{t-1})$ for all $d_t \in \mathbb{D}, t \in \mathcal{T}$. If demand is continuous, then $P(\cdot | d_{t-1})$ is the conditional density function.

**Assumption 2.** The demand is non-negative and bounded, i.e. there exists $D_{\text{max}} < \infty$ such that $0 \leq D_t \leq D_{\text{max}}$ for all $t \in \mathcal{T}$. If the demand is continuous, we further assume the conditional densities $f(\cdot | d_{t-1})$ for all $t \in \mathcal{T}$ are Lipschitz continuous. More specifically, there exists $L_d < \infty$ such that

$$|f(d_t | d_{t-1}) - f(d_t' | d_{t-1}')| \leq L_d \|d_{t-1} - d_{t-1}'\|, \quad \forall d_{t-1}, d_{t-1}', d_t \in \mathbb{D}, t \in \mathcal{T}.$$  

**Assumption 3.** The system has the option to expand/contract the installed capacity from $K_{t-1}$ to $K_t$ at the end of each period $t \in \mathcal{T}$, and capacity adjustments are instantaneous.
**Assumption 4.** The expansion cost and salvage value are linear with respect to the capacity, and the per unit expansion cost is not smaller than the per unit salvage value.

Assumption 2 is standard since real-world demands are always finite and their variation from one period to the next does not surge to infinity. In addition, we assume that the lead time of capacity adjustment (compared to the length of each time period) is negligible in our strategic capacity investment problem. However, the lead time for capacity adjustment in some industries can be long. For example, in the semiconductor industry, the lead time for purchasing machines can vary from six to eighteen months (Truong and Roundy 2011).

In this paper, we consider an MCIP with linear expansion cost and salvage value, but the proposed method can also solve more general cases whose expansion cost/salvage value are nonlinear. In addition, the resale value of an asset is usually smaller than its purchase price, so the second statement of Assumption 4 is realistic. The notation for our model is summarized in Table 1.

In each period, we can allocate the realized demand $d_t$ to the facilities, given the constraints of the currently installed capacity $K_{t-1}$. A penalty with unit cost $b_{it}$ is incurred if demand from customer $i$ is unsatisfied. Denote $z_{int}$ as the amount of demand allocated from customer $i$ to facility $n$ in time period $t$ and $r_{int}$ as its corresponding revenue. Denote the operating profit $\Pi_t(K_{t-1},d_t)$ in time $t \in T$, it is given by:

$$
\Pi_t(K_{t-1},d_t) = \max \sum_{i \in I} \sum_{n \in N} r_{int} z_{int} - \sum_{i \in I} b_{it} \left( d_{it} - \sum_{n \in N} z_{int} \right)
$$

(1)

s. t. \[
\sum_{i \in I} z_{int} \leq K_{n(t-1)}, \quad \forall n \in N; \quad (2)
\]

\[
\sum_{n \in N} z_{int} \leq d_{it}, \quad \forall i \in I, \quad (3)
\]

\[
z_{int} \geq 0, \quad \forall i \in I, n \in N. \quad (4)
\]

In the above, $z_{int}$ for all $i, n, t$ are the decision variables that allocate realized demands to the installed facilities and the objective function (1) is to maximize the current rewards, which consist of the revenues and the penalty for unsatisfied demands. Constraints (2) and (3) are capacity and
Table 1  Notations for the multi-stage MCIP

| **Indices and sets**                        |
|-------------------------------------------|
| $i$ Index for customers                   |
| $n$ Index for facility                    |
| $t$ Index for time period                 |
| $\mathcal{I}$ Set of customers, $i \in \mathcal{I}$, and $|\mathcal{I}| = I$ |
| $\mathcal{N}$ Set of facilities, $n \in \mathcal{N}$, and $|\mathcal{N}| = N$ |
| $\mathcal{T}$ Set of time periods, $t \in \mathcal{T}$, and $|\mathcal{T}| = T$ |

| **Parameters**                             |
|-------------------------------------------|
| $d_{it}$ Amount of demand generated from customer $i$ in time $t$; the vector form is denoted as $d_t = (d_{1t}, \ldots, d_{it})$ for all $t \in \mathcal{T}$ |
| $\gamma$ Discount factor of time value of money, $0 < \gamma < 1$ |
| $r_{int}$ Unit revenue from satisfying customer $i$ with facility $n$ in time $t$ |
| $b_{it}$ Unit penalty cost for unsatisfied customer $i$ in time $t$ |
| $q_{nt}^+$ Coefficient parameters of per unit expansion cost of facility $n$ in time $t$ |
| $q_{nt}^-$ Coefficient parameters of per unit salvage value of facility $n$ in time $t$ |

| **Variables**                              |
|-------------------------------------------|
| $K_{nt}$ Capacity of facility $n$ in time $t$; the vector form is $K_t = (K_{1t}, \ldots, K_{Nt})$ |
| $K_{n0}$ Initial capacity of facility $n$; the vector form is $K_0 = (K_{10}, \ldots, K_{N0})$ |
| $z_{int}$ Amount of demand allocated from customer $i$ to facility $n$ in time $t$ |

Demand constraints respectively. Note that the allocation decisions $z_{int}$ depend on the current state $(K_{i-1}, d_t)$ only and do not affect future activity.

Denote $c_t (K_{i-1}, K_t)$ as the capacity adjustment costs in time $t \in \mathcal{T}$. Denote $q_{nt}^+$ and $q_{nt}^-$ as the unit expansion cost and unit salvage value for facility $n \in \mathcal{N}$ respectively. According to Assumption
4, we have \( q^+_{nt} \geq q^-_{nt} \) for all \( n \in N, t \in T \). The cost function is then

\[
c_t(K_{t-1}, K_t) \triangleq \sum_{n \in N} \max \left\{ -q^-_{nt} \left( K_{n(t-1)} - K_{nt} \right), q^+_{nt} \left( K_{nt} - K_{n(t-1)} \right) \right\},
\]

which is convex in \( K_t \).

Based on the aforementioned assumptions, the MCIP can be modeled as an MDP. The state in time period \( t \in T \) can be represented by a two tuple \((K_{t-1}, d_t)\)—i.e. the installed capacity in time \( t - 1 \) and the realized demands—and the action is the adjusted capacity \( K_t \). The state space of our problem, therefore, is \( S \triangleq \{(K, d) \in \mathbb{K} \times \mathbb{D}\} \), and the action space is \( \mathbb{K} \) for all \( t \in T \). Denote \( K_t : S \rightarrow \mathbb{K} \) as a Markov decision rule for time \( t \in T \). We denote the class of Markov policies as \( K \triangleq \{(K_0, \ldots, K_T)\} \).

Without loss of generality, we assume the initial state for the MCIP is \((0_N, d_0)\), and the system salvages all installed capacity at the end of period \( T \) so \( K_T \equiv 0_N \). MCIP can then be formulated as the following dynamic optimization problem:

\[
\max_{(K_0, \ldots, K_T) \in K} -c_0(0_N, K_0) + \mathbb{E} \left[ \sum_{t \in T} \gamma^t (\Pi_t(K_{t-1}, D_t) - c_t(K_{t-1}, K_t)) \right]
\]

where \( 0 < \gamma < 1 \) is the discount factor. The objective of the above problem is to find an optimal policy, i.e. \((K_0^*, \ldots, K_T^*)\), such that the expected total rewards are maximized. We can solve Problem (5) via its dynamic programming equations.

**Theorem 1.** Let \( V_{T+1}(\cdot) \equiv 0 \) and define the following dynamic equations for all \((K_{t-1}, d_t) \in S\):

\[
V_t(K_{t-1}, d_t) = \max_{K_t \in \mathbb{K}} \left\{ \Pi_t(K_{t-1}, d_t) - c_t(K_{t-1}, K_t) + \gamma \mathbb{E} \left[ V_{t+1}(K_t, D_{t+1}) \right] \right\}, \quad t \in T,
\]

\[
V_0(0_N, d_0) = \max_{K_0 \in \mathbb{K}} \left\{ -c_0(0_N, K_1) + \gamma \mathbb{E} \left[ V_1(K_0, D_2) \right] \right\}, \quad t = 0.
\]

Given Assumption 1, Eqs. (6)–(7) recover the optimal policy of Problem (5).

Above, we model MCIP as a finite horizon MDP with a finite action space. This problem can be solved by DP-based algorithms, such as value iteration (VI), but VI is not applicable to MCIP when the dimensions of the state/action space are high. More specifically, if demands are discrete, the complexity of VI is \( O \left( |S|^2 \times |\mathbb{K}| \times T \right) \), where \( |S| \) is of \( I + N \) dimensions and \( \mathbb{K} \) is of \( N \) dimensions.
For example, consider a system with four customers and three facilities and $T = 10$. If $K_{\text{max}} = 9$ and demands of each customer are integer values ranging from 1 to 10, then the complexity of VI is $(10^3 \times 10^4)^2 \times 10^3 \times 10 = 10^{18}$. Therefore, VI is intractable even for a medium size problem. If the demands are continuous, the state space is infinite and exact VI cannot be done.

3.1. The FVI Algorithm

Since the state space $S$ is large or continuous, evaluating the exact value functions in Eqs. (6)–(7) is intractable because of the curse of dimensionality. To respond to this challenge, we will use FVI to fit the value functions by a finite number of samples generated from the state space. The required number samples is generally much smaller than the original state space. Then, we can approximately solve the Bellman equations in Eqs. (6)–(7) by using the approximated value functions instead of the exact ones.

Define $S_1 \triangleq \{1, \ldots, S_1\}$ and $S_2 \triangleq \{1, \ldots, S_2\}$ as the sets of indices for the samples generated from the state space and from the state transitions, respectively. First, a set of state samples $(K_{t-1}^s, d_t^s)_{s \in S_1}$ is drawn from $S$ in the last period $t = T$, and their values $\hat{V}_T (K_{t-1}^s, d_t^s)$ are calculated according to Eq. (6). Then, a set of samples, i.e. $((K_{t-1}^s, d_t^s), \hat{V}_T (K_{t-1}^s, d_t^s))_{s \in S_1}$, can be generated. Denote $\hat{V}_t (\cdot; w)$ as the parametric approximate value functions given adjustable parameters $w \in W$, where $W \subset \mathbb{R}^{\text{dim}(W)}$ and dim $(W)$ is finite (a simple example is to approximate the value functions via linear basis functions, i.e. $\hat{V}_t (x; w) = w^\top x + w_0$). Then, $\hat{V}_t (\cdot; w)$ can be trained by solving the following regression problem:

$$
\hat{w}_T = \arg \min_{w \in W} \frac{1}{S_1} \sum_{s \in S_1} \left( \hat{V}_T (K_{t-1}^s, d_t^s; w) - \hat{V}_T (K_{t-1}^s, d_t^s) \right)^2 + \frac{\beta}{2} w^\top w,
$$

where $\beta > 0$ is the regularization parameter (regularizing the objective function can sometimes improve the generalization of the function fitting and avoid overfitting). Subsequently, the algorithm proceeds backwards in time $t$. In time $t \in T \setminus \{T\}$, we draw $S_1$ number of samples of current states. Since the transitions of demands are independent of the actions, for each state sample $(K_{t-1}^s, d_t^s)$, we generate the future transitions $d_{t+1}^s$ given $d_t^s$ via Monte Carlo simulations; that is

$$
d_{t+1}^s \sim P (\cdot | d_t^s), \quad \forall s \in S_1, s' \in S_2, t \in T \setminus \{T\}.
$$
Then, we can calculate the estimated values \( \hat{V}_t (K_{t-1}^s, d_t^s) \) given the trained function \( \tilde{V}_{t+1} (\cdot; \hat{w}_{t+1}) \), and solve \( \hat{V}_t (\cdot; \hat{w}_t) \). The detailed procedure of the algorithm is summarized below.

**Algorithm 1** The FVI algorithm

- **Step 0:** Initialize \( S_1, S_2 \), initial state \((0_N, d_0)\), and the MDP parameters. Set \( t \leftarrow T \).
- **Step 1:** Draw samples \((K_{t-1}^s, d_t^s)\) \( s \in S_1 \) independently from the state space \( S \) and generate samples of future transitions \((d_{t+1}^s(s))_{s \in S_2} \) given each \( d_t^s \) for all \( s \in S_1 \).
- **Step 2:** Compute for \( s \in S_1 \),
  \[
  \hat{V}_t (K_{t-1}^s, d_t^s) = \begin{cases} 
  \max_{K_t \in \mathbb{K}} \left[ \Pi_t (K_{t-1}^s, d_t^s) - c_t (K_{t-1}^s, K_t) + \gamma \frac{1}{|S_2|} \sum_{s' \in S_2} \tilde{V}_{t+1} (K_t, d_{t+1}^s(s); \hat{w}_{t+1}) \right], & 1 \leq t < T, \\
  \Pi_t (K_{t-1}^s, d_t^s) - c_t (K_{t-1}^s, 0_N), & t = T.
  \end{cases}
  \]
  \( (8) \)
- **Step 3:** Given samples \( (K_{t-1}^s, d_t^s), \hat{V}_t (K_{t-1}^s, d_t^s) \) \( s \in S_1 \), fit the approximate value functions
  \[
  \hat{w}_t = \arg \min_{w \in W} \frac{1}{|S_1|} \sum_{s \in S_1} (\hat{V}_t (K_{t-1}^s, d_t^s) - \hat{V}_t (K_{t-1}^s, d_t^s))^2 + \frac{\beta}{2} w^\top w, \ t \in T.
  \]
  \( (9) \)
- **Step 4:** If \( t > 0 \), set \( t \leftarrow t - 1 \) and go to Step 1; otherwise, terminate and return
  \[
  \hat{V}_0 (0_N, d_0) = \max_{K_0 \in \mathbb{K}} \left[ -c_0 (0_N, K_0) + \gamma \frac{1}{|S_2|} \sum_{s' \in S_2} \hat{V}_1 (K_0, d_{t+1}'(s'); \hat{w}_1) \right].
  \]

There are two questions that need to be answered about the above FVI algorithm. First, what is the appropriate approximator such that \( \hat{V}_t (\cdot; w) \) can fit the true function \( V_t (\cdot) \) with arbitrary precision? Second, the action selection problem, i.e. Eq. (8), involves a multi-dimensional action space \( \mathbb{K} \) such that it is difficult to solve via the enumeration method; how can we speed up this algorithm? We can answer both questions. We discuss the choice of approximator in Section 4 and then introduce a decomposition algorithm to accelerate Problem (8) in Section 5.

### 4. Neural Network-based Fitted Value Iteration Algorithm

Munos and Szepesvári (2008, Theorem 2 & Corollary 4) have shown that FVI can achieve an \( \epsilon \)-optimal solution with probability \( \delta \) when: (i) the functional family used for approximation is sufficiently rich, and (ii) the sample complexity increase polynomially in the scale of the problem instance. In particular, we want to show that the value functions of our MDP are Lipchitz
functions. Then, we will show that our approximator is rich enough to handle Lipschitz functions.

We first verify that the value functions of MCIP are Lipchitz, and then we discuss the choice of approximators such that FVI is able to derive $\epsilon$-optimal solution with high probability.

### 4.1. Fitting the Value Function

The value functions $V_t(\cdot)$ for all $t \in T$ are defined over $S$, which is not connected since $K$ is finite. In addition, if demand is discrete, then the state space is finite and discrete. To have a better understanding of the structure of the value functions, we wish to extend $K$ to its smallest connected superset. Then, we will construct a set of extended value functions.

First we define
\[
\bar{S} \triangleq \{ \bar{K} \times \bar{D} \} \quad \text{and} \quad \bar{K} \triangleq \{ K \in \mathbb{R}^N | 0 \leq K \leq K^{\max} \},
\]
where $S \subset \bar{S}$. Note that $D$ can be either continuous or discrete in the above definitions. Now, we may define extended value functions $\bar{V}_t(\cdot): \bar{S} \rightarrow \mathbb{R}$ for all $t \in T$.

The study of $\bar{V}_t(\cdot)$ is more amenable since $\bar{K}$ is a connected set.

The dynamic programming equations for the extended value functions at $(K_{t-1}, d_t) \in \bar{S}$ are given by
\[
\bar{V}_{T+1}(\cdot) \equiv 0 \quad \text{and} \quad \bar{V}_t(K_{t-1}, d_t) \triangleq \max_{K_t \in \bar{K}} \{ \Pi_t(K_{t-1}, K_t) - c_t(K_{t-1}, K_t) + \gamma E[\bar{V}_{t+1}(K_t, D_{t+1})|d_t] \}, \quad t \in T, \tag{10}
\]
\[
\bar{V}_0(0_N, d_0) \triangleq \max_{K_0 \in \bar{K}} \{ -c_0(0_N, K_1) + \gamma E[V_1(K_1, D_2)|d_0] \}, \quad t = 0. \tag{11}
\]

Note that the values of $\Pi_t(\cdot)$ and $c_t(\cdot)$ are attained given any $(K_{t-1}, d_t) \in \bar{S}$.

Next we show that the value functions $V_t(\cdot)$ can be recovered from the extended functions $\bar{V}_t(\cdot)$ on $S$.

**Proposition 1.** $\bar{V}_t(K_{t-1}, d_t) = V_t(K_{t-1}, d_t)$ for all $(K_{t-1}, d_t) \in S, t \in T$.

**Proof.** According to the definition of $\bar{V}_T(\cdot)$, we have $\bar{V}_T(\cdot) = V_T(\cdot)$ at the points of $(K_{T-1}, d_T) \in S$. Suppose we have $\bar{V}_{t+1}(\cdot) = V_{t+1}(\cdot)$ for all $(K_t, d_{t+1}) \in S$. Since the demand transitions are independent of $K_{t-1}$ and $K_t$, we have $E[\bar{V}_{t+1}(K_t, D_{t+1})|d_t] = E[V_{t+1}(K_t, D_{t+1})|d_t]$ for all $K_t \in K$. 

Therefore, $\tilde{V}_t(K_{t-1}, d_t) = V_t(K_{t-1}, d_t)$ for all $(K_{t-1}, d_t) \in \mathbb{S}$ according to Eq. (6) and Eq. (10). The result then follows by backward induction. ☐

We now show that the extended value functions of MCIP, i.e. $\{\tilde{V}_t(\cdot)\}_{t \in T}$, are bounded and Lipchitz. First, we show that the reward function is bounded by a constant

$$v_{\text{max}} \triangleq \sum_{i \in I} \max_{t \in T, n \in N} (r_{int} + b_{it}) D_i^{\text{max}} + \sum_{n \in N} \max_{t \in T} q_{nt}^+ K_n^{\text{max}}.$$  

**Lemma 1.** We have $v_{\text{max}} \geq |\Pi_t(K_{t-1}, d_t) - c_t(K_{t-1}, K_t)|$ for all $(K_{t-1}, d_t) \in \tilde{\mathbb{S}}, K_t \in \mathbb{K}, t \in T$.

**Proof.** As $\tilde{\mathbb{S}}$ is a bounded set and $d_t \leq D_t^{\text{max}}$ for all $t \in T$, an upper and lower bound on $\Pi_t(K_{t-1}, d_t)$ follows if we have unlimited or no capacity:

$$- \sum_{i \in I} \max_{t \in T} b_{it} D_i^{\text{max}} \leq \Pi_t(K_{t-1}, d_t) \leq \sum_{i \in I} \max_{t \in T, n \in N} r_{int} D_i^{\text{max}}.$$  

Therefore,

$$|\Pi_t(K_{t-1}, d_t)| \leq \sum_{i \in I} \max_{t \in T, n \in N} (r_{int} + b_{it}) D_i^{\text{max}}.$$  

Similarly, bounds on $c_t(K_{t-1}, K_t)$ follow by assuming the capacity is changed from zero to $K_t^{\text{max}}$ or the reverse:

$$- \sum_{n \in N} \max_{t \in T} q_{nt}^- K_n^{\text{max}} \leq c_t(K_{t-1}, K_t) \leq \sum_{n \in N} \max_{t \in T} q_{nt}^+ K_n^{\text{max}}.$$  

Since we have assumed $q_{nt}^- \leq q_{nt}^+$, it follows that

$$|c_t(K_{t-1}, K_t)| \leq \sum_{n \in N} \max_{t \in T} q_{nt}^+ K_n^{\text{max}}.$$  

Then, for all $t \in T, K_t \in \mathbb{K}$, we have

$$|\Pi_t(K_{t-1}, d_t) - c_t(K_{t-1}, K_t)| \leq |\Pi_t(K_{t-1}, d_t)| + |c_t(K_{t-1}, K_t)|$$

$$\leq \sum_{i \in I} \max_{t \in T, n \in N} (r_{int} + b_{it}) D_i^{\text{max}} + \sum_{n \in N} \max_{t \in T} q_{nt}^+ K_n^{\text{max}}$$

$$= v_{\text{max}},$$

which concludes the proof. ☐
Next, we show that the extended value functions are $L_v$–Lipschitz where
\[
L_v \triangleq \left( 2 + \sum_{\tau=0}^{T} \gamma^{\tau+1} L_d \right) v_{\text{max}}.
\]
This property is extremely important because it allows us to show that our neural net approximation architecture is rich enough.

**Proposition 2.** The functions $\bar{V}_t (\cdot)$ for all $t \in T$ in Eq. (10) are Lipschitz, i.e. $|\bar{V}_t (x) - \bar{V}_t (x')| \leq L_v \|x - x'\|$ for all $x, x' \in \mathbb{S}$ and $t \in T$.

**Proof.** We only provide the proof for the continuous demand case with density function $f (\cdot | d_t)$; the discrete case can be proved similarly. To simplify the notation, we denote the state variable $(K_{t-1}, d_t)$ as $x$ and the action $K_t$ as $a$. Firstly, we show that $\bar{V}_t (x)$ for all $t \in T$ are bounded. According to Lemma 1, we have $v_{\text{max}} \geq |\Pi_t (x) - c_t (x, a)|$ for all $x \in \mathbb{S}, a \in \mathbb{K}, t \in T$. For $t = T$, we have $|\bar{V}_T (x)| = |\Pi_T (x) - c_T (x, a)| \leq v_{\text{max}}$, for all $x \in \mathbb{S}, a \in \mathbb{K}$. Note that $P (\cdot | x)$ is independent with $a \in \mathbb{K}$, and we have
\[
|\bar{V}_t (x)| \leq \max_{a \in \mathbb{K}} \left| \Pi_t (x) - c_t (x, a) + \gamma \int_{x' \in \mathbb{S}} \bar{V}_{t+1} (x') dP (x' | x) \right| \leq \left( \sum_{\tau=0}^{T} \gamma^{\tau} \right) v_{\text{max}}, \quad \forall x \in \mathbb{S}, t \in T.
\]
Then, we prove the Lipschitz condition for $\bar{V}_t (\cdot)$. Firstly, for all $x, x' \in \mathbb{S}, t \in T$, compute
\[
\max_{a \in \mathbb{K}} (\Pi_t (x) - c_t (x, a) - \Pi_t (x') + c_t (x', a)) \leq \max_{a \in \mathbb{K}} |\Pi_t (x) - c_t (x, a) - \Pi_t (x') + c_t (x', a)| \leq 2v_{\text{max}} \|x - x'\|.
\]
For $t = T$, we have
\[
|\bar{V}_T (x) - \bar{V}_T (x')| = |\Pi_T (x) - \Pi_T (x') + c_T (x, a) - c_T (x', a)| \leq 2v_{\text{max}} \|x - x'\|.
\]
In general, for $t < T$, we have
\[
|\bar{V}_t (x) - \bar{V}_t (x')| \leq \max_{a \in \mathbb{K}} (\Pi_t (x) - c_t (x, a) + \mathbb{E} \bar{V}_{t+1} (y | x, a)) - \max_{a' \in \mathbb{K}} (\Pi_t (x') - c_t (x', a') + \mathbb{E} \bar{V}_{t+1} (y' | x', a'))
\leq \max_{a \in \mathbb{K}} (\Pi_t (x) - c_t (x, a) - \Pi_t (x') + c_t (x', a)) + \left| \max_{a \in \mathbb{K}} [\mathbb{E} \bar{V}_{t+1} (y | x, a) - \mathbb{E} \bar{V}_{t+1} (y' | x', a')] \right|
\leq 2v_{\text{max}} \|x - x'\| + \gamma \left( \max_{a \in \mathbb{K}} \left| \int_{y} \mathbb{E} \bar{V}_{t+1} (y) (P (y | x) - P (y' | x')) dy \right| \right)
\leq 2v_{\text{max}} \|x - x'\| + \gamma \left( \sum_{\tau=0}^{T} \gamma^{\tau} \right) v_{\text{max}} \int |f (y | x) - f (y | x')| dy
\leq \left( 2 + \sum_{\tau=0}^{T} \gamma^{\tau+1} L_d \right) v_{\text{max}} \|x - x'\|.
\]
using the fact that $\bar{V}_t(\cdot)$ for $t \in T$ are bounded according to Assumption 2. Thus, there exists $L_v = \left( 2 + \sum_{t=0}^{T} \gamma^{t+1} L_d \right) v_{\text{max}}$ such that $\bar{V}_t(\cdot)$ are Lipschitz functions.

If demands are discrete, we can replace the integration in Eq. (12)-(14) by summation, and the result holds trivially since demands automatically are bounded in this case. □

FVI proceeds via backward induction: the value function in the last period is solved first, then approximated, and then the remaining value functions are calculated by going backwards in time. If the approximation in the last period is poor, the error can be passed on to the approximation in earlier stages. In the next result, we use backward induction to understand the structure of $\bar{V}_t(\cdot)$.

**Proposition 3.** The extended value functions, i.e. Eqs. (10)-(11), have the following properties:

(i) $\bar{V}_T(\cdot)$ is a piecewise linear function;

(ii) $\bar{V}_t(\cdot)$ for all $t \in T \setminus \{T\}$ are piecewise linear functions if $\mathbb{D}$ is finite.

**Proof.** First, we know that piecewise linearity is preserved under finite summation and the max/min of a finite collection. Now, observe that $\Pi_t (K_{t-1}, d_i)$ is a piecewise linear function when defined over $(K_{t-1}, d_i) \in \bar{S}$. This result follows by transforming $\Pi_t (K_{t-1}, d_i)$ into its dual. Since $\Pi_t (K_{t-1}, d_i)$ is a linear programming problem and the optimal value is finite and attained for all $(K_{t-1}, d_i) \in \bar{S}$, strong duality holds (see e.g. (Bertsimas and Tsitsiklis 1997, Chapter 4)). Thus, we have

$$\Pi_t (K_{t-1}, d_i) = \min_{\mu_n, \lambda_i \geq 0} \sum_{i \in I} (b_{it} - \lambda_i) d_{it} - \sum_{n \in N} \mu_n K_{n(t-1)}$$

$$\text{s. t. } (\mu_n + \lambda_i - r_{int} - b_{it}) \geq 0, \ \forall i \in I, n \in N.$$

Let $\Lambda$ denote the feasible set of $(\mu, \lambda)$ for the above problem, where $\lambda = (\lambda_i)_{i \in I}$ and $\mu = (\mu_n)_{n \in N}$. Since the dual is also a linear programming problem, the optimal solutions $(\mu^*, \lambda^*)$ can be chosen from the extreme points of their feasible regions (Bertsimas and Tsitsiklis 1997, Chapter 3), and so the above problem is equivalent to

$$\min_{(\mu, \lambda) \in \text{ext}(\Lambda)} \sum_{i \in I} (b_{it} - \lambda_i) d_{it} - \sum_{n \in N} \mu_n K_{n(t-1)}.$$
Therefore, $\Pi_t(K_{t-1}, d_t)$ is piecewise linear and concave in $K_{t-1} \in \mathbb{K}$ since it is the min of a finite collection of linear functions. Since $\bar{V}_T(K_{T-1}, d_T) = \Pi_T(K_{T-1}, d_T) - c_T(K_{T-1}, 0_N)$ and $c_T(\cdot)$ is convex in $K_{T-1}$, $\bar{V}_T(K_{T-1}, d_T)$ is piecewise linear for all $(K_{T-1}, d_T) \in \bar{S}$ and concave in $K_{T-1}$.

To prove (ii), as we have shown that $\bar{V}_T(\cdot)$ is piecewise linear and concave, by backward induction, we have

$$
\bar{V}_t(K_{t-1}, d_t) = \max_{K_t \in \mathbb{K}} \left\{ \Pi(K_{t-1}, d_t) - c(K_{t-1}, K_t) + \gamma \mathbb{E} \left[ \bar{V}_{t+1}(K_{t}, D_{t+1}) | d_t \right] \right\}, \ \forall t \in T \setminus \{T\}.
$$

Since $\bar{V}_{t+1}(K_t, D_{t+1})$ is piecewise linear and $D_{t+1} \in \mathbb{D}$ is finite, $\mathbb{E} \left[ \bar{V}_{t+1}(K_{t}, D_{t+1}) | d_t \right]$ is piecewise linear as it is a finite sum of piecewise linear functions. Therefore, $\bar{V}_t(K_{t-1}, d_t)$ is piecewise linear as the max of a finite set of piecewise linear functions. □

**Remark 1.** Note that $\bar{V}_t(\cdot, d_t)$ for all $t \in T \setminus \{T\}$ are non-concave in $K_{t-1} \in \mathbb{K}$ given any $d_t \in \mathbb{D}$. As $\bar{V}_T(\cdot, d_T)$ is concave in $K_{T-1} \in \mathbb{K}$ given any $d_T \in \mathbb{D}$, $\mathbb{E} \left[ \bar{V}_t(\cdot, D_t) | d_{T-1} \right]$ is concave in $K_{T-1} \in \mathbb{K}$ given $d_{T-1}$ (see the proof in Proposition 3), $\bar{V}_{T-1}(\cdot, d_{T-1})$ may be non-concave as it is a finite max of concave functions, and the result follows by backward induction.

According to Proposition 3(i), we know that $\bar{V}_T(\cdot)$ is a piecewise linear function if the capacity is defined over a connected space $\mathbb{K}$. However, for $t \in T \setminus \{T\}$, the value functions $\bar{V}_t(\cdot)$, according to Proposition 3(ii), are piecewise linear only when the demand is discrete. Fortunately, if the demands are continuous, we can use Monte-Carlo simulation to generate finitely many samples of future transitions to approximate the expectation in Eqs. (6)–(7). In this setting, the approximate value functions are still piecewise linear. However, given Remark 1, we may need to solve non-convex optimization problems in each $t \in T \setminus \{T\}$.

### 4.2. Two-layer Neural Network with ReLU

To solve the MCIP, we approximate the value functions of the problem by using neural networks with piecewise linear activation functions. Neural networks are powerful approximators, and can incorporate structure of the target function (Jain et al. 1996). In our case, we will use two-layer
neural networks. As we will see later in this section, a two-layer network is powerful enough to approximate our value functions arbitrarily well.

A two-layer neural network consists of inputs, one hidden layer for intermediate computations, and an output. Let $\mathcal{J} \triangleq \{1, \ldots, J\}$ index the neurons in the hidden layer of our network. The general form of a two-layer neural network is then

$$\Gamma (x) = \sum_{j \in \mathcal{J}} w_j \Psi_j \left( u_j^T x + u_{0j} \right) + w_{0j},$$

where $(u_j, u_{0j})$ and $(w_j, w_{0j})$ are the adjustable weights of the input layer and the hidden layer respectively, and $\Psi_j (\cdot)$ is the activation function for neuron $j$. In MCIP, the inputs of the networks are the states $(K_{t-1}, d_t) \in \mathcal{S}$ and the outputs are the approximate values $\tilde{V}_t (\cdot)$. Based on Proposition 3, we choose the activation functions $\Psi_j (\cdot)$ to be ReLU, which are themselves piecewise linear

$$\Psi_j \left( u_j^T x + u_{0j} \right) = \max \left\{ u_j^T x + u_{0j}, 0 \right\}.$$

The architecture of the neural network is shown in Figure 1.

![Figure 1](image.png)

Figure 1 (a) the plot of ReLU and (b) the architecture of the two-layer neural network

The adjustable weights for the input of neuron $j$ in $\tilde{V}_t (\cdot)$ include $u_{0jt}$ and an $I + N$ vector of adjustable coefficients

$$u_{jt} = (u_{1jt}, \ldots, u_{Njt}, u_{(N+1)jt}, \ldots, u_{(N+I)jt}), \quad \forall j \in \mathcal{J}.$$  

Denote $u_t = (u_{0jt}, u_{jt})_{j \in \mathcal{J}}$ as the vector of all adjustable weights for the input layer and

$$w_t = (w_{0t}, w_{1t}, \ldots, w_{Jt})$$
as a $J + 1$ vector of the adjustable parameters for the output of the hidden layer. The neural network at time $t \in \mathcal{T}$ can then be represented by a function given the input $(K_{t-1}, d_t) \in \mathbb{S}$ and the adjustable weights $(u_t, w)$:

$$
\tilde{V}_t(K_{t-1}, d_t; u_t, w_t) = \sum_{j \in \mathcal{J}} w_{jt} \max \left\{ u_{jt}^T (K_{t-1}, d_t) + u_{0jt}, 0 \right\} + w_{0t}, \ \forall t \in \mathcal{T}.
$$

(15)

We run NN-FVI on MCIP using the approximate value functions Eq. (15).

4.3. Consistency of the NN-FVI Algorithm

In this section, we show that our customized neural network is powerful enough to approximate the target functions. In particular, we show that NN-FVI is consistent.

Recall that, according to Proposition 2, the extended value functions $\{\tilde{V}_t\}_{t \in \mathcal{T}}$ are Lipschitz with constant $L_v$. We define

$$
\text{Lip}(L_v) \triangleq \{ f : \bar{\mathbb{S}} \mapsto \mathbb{R} \mid |f(x) - f(y)| \leq L_v \| x - y \|, \forall x, y \in \bar{\mathbb{S}} \}
$$

to be the class of $L_v$-Lipschitz functions. The approximation power of a function set $\mathcal{F}$ on $\bar{\mathbb{S}}$ can be measured with respect to $\text{Lip}(L_v)$:

$$
\mathcal{E}_p(\text{Lip}(L_v), \mathcal{F}) \triangleq \sup_{g \in \text{Lip}(L_v)} \inf_{f \in \mathcal{F}} \| g - f \|_p.
$$

Since $\tilde{V}_t(\cdot) \in \text{Lip}(L_v)$ for all $t \in \mathcal{T}$, if $\mathcal{E}_p(\text{Lip}(L_v), \mathcal{F})$ is small, then the approximation error between $\mathcal{F}$ and $\tilde{V}_t(\cdot)$ is also small. If, for any $\alpha$ and $L$, a sequence $\{\mathcal{F}_{j'}\}$ satisfies $\lim_{j' \to \infty} \mathcal{E}_p(\text{Lip}(L_v), \mathcal{F}_{j'}) = 0$, then it is universal (Antos et al. 2007). This condition implies that the inherent approximation error between the two function spaces converges to zero as the function set $\{\mathcal{F}_{j'}\}$ becomes “richer”.

The richness of a neural network is quantified by the number of layers and weights (Bartlett et al. 2017). The approximation power of our customized two-layer network increases with the number of neurons. We have the following main result about the quality of our two-layer neural network approximation.
Lemma 2. The two-layer neural network with ReLU is a universal approximator.

Sketch of Proof. We only provide a sketch of proof here, and refer the interested readers to (Hornik et al. 1989) or (Sonoda and Murata 2015) for details. According to (Hornik et al. 1989, Corollary 2.2), a two-layer neural network is universal if the activation function of the network, denoted as \( \Psi(x) \), is a squashing function. The squashing function should satisfy the following properties according to (Hornik et al. 1989, Definition 2.3): \( \Psi(x) \) is non-decreasing, \( \lim_{x \to \infty} \Psi(x) = 1 \), and \( \lim_{x \to -\infty} \Psi(x) = 0 \).

ReLU does not satisfy the second property of the squashing function but we can construct an equivalent network with squashing activation functions via a linear combination of two ReLUs. Suppose we have a ReLU network \( \Gamma(x) \) with sufficient neurons, and we assume the number of neurons \( J \) is even without loss of generality. For neuron \( j \in \{1, \ldots, J/2\} \), we pick out \( j' = J/2 + j \) and specify its adjustable weights such that \( u_{j'} = u_j, \ u_{0j'} = u_{0j} - 1, \) and \( w_{j'} = -w_j \). Then, we construct an activation function via a linear combination of these two neurons:

\[
\hat{\Psi}_j(x) = \max \{0, u_j^T x + u_{0j}\} - \max \{0, u_{j'}^T x + u_{0j} - 1\}.
\]

Given \( \hat{\Psi}_j(x) \), a new network \( \hat{\Gamma}(x) \) with \( J/2 \) neurons is thus constructed. Apparently, \( \hat{\Gamma} \) is universal as \( \hat{\Psi}_j(x) \) are squashing functions: \( \hat{\Psi}(x) \) is non-decreasing, \( \lim_{x \to \infty} \hat{\Psi}(x) = 1 \), and \( \lim_{x \to -\infty} \hat{\Psi}(x) = 0 \). Since the solutions of the weights of \( \hat{\Gamma} \) are a subset of those of \( \Gamma \), there always exists network \( \Gamma \) that is equivalent to \( \hat{\Gamma} \). Thus, the two-layer network with ReLU is universal. \( \square \)

Denote \( \mu \) as a distribution function defined over \( S \), which is used to generate samples from the state space. Given Assumption 2, Proposition 2, and Lemma 2, the consistency of NN-FVI algorithm follows from (Munos and Szepesvári 2008, Corollary 4). The result is presented below and its detailed proof appears in Appendix. The idea of the proof is to check the conditions of (Munos and Szepesvári 2008, Corollary 4).

Theorem 2. Consider an MDP formulated by Eqs. (6)–(7) and satisfying Assumptions 1 and 2. For any \( \epsilon > 0, \delta \in (0,1), \) and \( 1 \leq p < \infty \), there exists an integer \( J_0 \) such that for any \( J \geq J_0 \) there
are \( S_1, S_2 \) that are polynomial in the quantities of the MDP such that \( \| \hat{V}_t - V_t \|_{p, \mu} \leq \varepsilon \) holds for all \( t \in T \) with probability at least \( 1 - \delta \).

**Remark 2.** Theorem 2 confirms the consistency of NN-FVI. However, the neural network training problem, e.g. Eq. (9) in Algorithm 1, is non-convex. Fortunately, it has been shown in experiments that as the size of the network increases, the chance of getting stuck in poor local minima decreases. In addition, when the network is large enough, finding the global minimum may be unnecessary as it often leads to overfitting (Choromanska et al. 2014).

### 5. Accelerate the Action Selection Procedure in NN-FVI

In NN-FVI, once the value function in time \( t+1 \) is fitted, one needs to solve Problem (8) (the action selection problem). Given that the capacity is finite, this problem can be solved by enumerating all possible actions (just the brute-force method). However, this method is subject to the curse of dimensionality. For example, if a system has five facilities and \( K_n^{\text{max}} = 10 \) for all \( n \in \mathcal{N} \), the complexity of brute-force action selection is \( 11^5 \). To address this issue, we solve the action selection problem as a two stage stochastic programming problem where we speed it up with a specialized decomposition algorithm. As the procedure is the same for each state sample in \( \mathcal{S}_1 \), with some abuse of notation, we suppress the dependence of the coefficients and parameters on the state sample. So, we just write \( d_{t+1}^s \) for all \( s \in \mathcal{S}_2 \) to denote samples of future transitions that are generated via Monte Carlo simulation given a specific \( d_t \).

#### 5.1. Formulation of the Action Selection Problem

For time period \( t \in T \setminus \{T\} \), suppose we have trained the neural network in time \( t+1 \) in Step 3 of Algorithm 1 and have its adjustable weights \( (u_{t+1}, w_{t+1}) \). Now, we need to solve Problem (8). Since \( \Pi_t (K_{t-1}, d_t) \) is constant given \( (K_{t-1}, d_t) \), it can be removed from the objective. Hence, for all \( t \in T \), Problem (8) is equivalent to

\[
\min_{K_t \in \mathbb{R}} c_t (K_{t-1}, K_t) + \gamma v_{t+1} (K_t, d_t),
\]  

(16)
where \( v_{t+1}(K_t, d_t) \) is the recourse function given action \( K_t \) and realized demand \( d_t \),

\[
v_{t+1}(K_t, d_t) \triangleq - \sum_{s \in \mathcal{S}_2} \tilde{V}_{t+1}(K_t, d_{t+1}; u_{t+1}, w_{t+1}).
\] (17)

This problem is now essentially a two-stage stochastic programming problem with a simple recourse function: the first-stage is to determine the capacity decisions, and the recourse function returns the expectation of the future costs based on the trained neural network in time \( t + 1 \).

Recall that the activation function of our neural network is ReLU. Problem (16) has two characteristics:

1. The recourse function is complete; that is, given any \( K_t \in \mathbb{K} \), the recourse is not empty as the neural network is defined over all \( \tilde{S} \) where \( \mathcal{S} \subset \tilde{S} \).

2. The recourse function may be non-convex in \( K_t \in \mathbb{R}^N \).

The possible non-convexity of the recourse function can be verified by transforming the function into its epigraph formulation; that is,

\[
\eta \geq \sum_{s \in \mathcal{S}_2} \sum_{j \in \mathcal{J}} \left[ -w_{j(t+1)} \max \left\{ u_{j(t+1)}^T \left( K_t, d_{t+1}^s \right) + u_{0j(t+1)}, 0 \right\} \right], \quad \forall K_t \in \tilde{K}, \eta \in \mathbb{R}.
\]

We see that the epigraph of the recourse function is obtained by summing up the epigraphs of the hidden layer outputs for all \( j \in \mathcal{J} \). As can be seen in Fig. 2, the epigraph of neuron \( j \) is convex if \( w_{j(t+1)} \leq 0 \); on the other hand, if \( w_{j(t+1)} > 0 \) then the epigraph is not convex. Note however that the sign of \( u_{j(t+1)} \) does not change the convexity of the epigraph.

![Figure 2](image)

*Figure 2*  Epigraphs of the output of neuron \( j \) when (a) \( w_{j(t+1)} \leq 0 \) and (b) \( w_{j(t+1)} > 0 \)
Based on these observations, we separate the neurons with positive and non-positive weights according to

\[ \mathcal{J}_{t+1}^+ = \{ j \in \mathcal{J} \mid w_j(t+1) > 0 \} \quad \text{and} \quad \mathcal{J}_{t+1}^- = \{ j \in \mathcal{J} \mid w_j(t+1) \leq 0 \}. \]

Then, we may define

\[ \text{epi}^+_t \triangleq \left\{ (K_t, \eta) \in \mathbb{R}^{N+1} \mid \eta^+ \geq \sum_{s \in \mathcal{S}_2} \sum_{j \in \mathcal{J}_{t+1}^+} \left[ -w_j(t+1) \max \left\{ u_j^T(K_t, d^s_{t+1}) + u_0(t+1), 0 \right\} \right] \right\}, \]

\[ \text{epi}^-_t \triangleq \left\{ (K_t, \eta) \in \mathbb{R}^{N+1} \mid \eta^- \geq \sum_{s \in \mathcal{S}_2} \sum_{j \in \mathcal{J}_{t+1}^-} \left[ -w_j(t+1) \max \left\{ u_j^T(K_t, d^s_{t+1}) + u_0(t+1), 0 \right\} \right] \right\}, \]

to be the epigraphs for the output of the neurons with positive/non-positive weights, respectively.

By changing the order of summation, Problem (16) (the action selection problem) given \((K_{t-1}, d_t)\) can be reformulated as

\[
\begin{align*}
\min_{K_t, \eta^+, \eta^-} & \quad c_t(K_{t-1}, K_t) + \gamma \left( \eta^+ + \eta^- \right) - \gamma w_0(t+1) \\
\text{s.t.} & \quad (K_t, \eta^+) \in \text{epi}^+_t(d_t), \\
& \quad (K_t, \eta^-) \in \text{epi}^-_t(d_t), \\
& \quad K_t \in \mathbb{K}.
\end{align*}
\]

We now define valid inequalities for the above problem.

**Definition 1.** An inequality \(\phi^T K_t - \eta + \phi_0 \leq 0\) of Problem (16) is valid for epigraph \(\text{epi}^*_t\), with * denoting + or −, if

\[ \text{epi}^*_t \subset \{ (K_t, \eta) \in \mathbb{R}^{N+1} \mid \phi^T K_t - \eta + \phi_0 \leq 0 \}. \]

In other words, a valid inequality is a constraint that contains the feasible region of the original problem. To solve Problem (16), we can decompose the problem and iteratively approximate the epigraph of its recourse function. Let \(m\) denote iterations of this decomposition algorithm, and let \(K_t^m\) denote the optimal solution of the first-stage problem in the \((m-1)\)th iteration. The decomposition procedure can be summarized as below:
1. To begin, we solve the first-stage problem and compute $K_{m}^m$ in the $(m - 1)$th iteration.

2. Then, we construct a valid inequality given $K_{m}^m$, add it into the first-stage problem, and update the first-stage decision in the $m$th iteration.

3. The algorithm is solved iteratively and the recourse functions are gradually approximated via the valid inequalities from below.

By implementing the decomposition algorithm, the first-stage of the action selection problem becomes a mixed integer linear programming problem with the same number of integer variables as the number of facilities, which is solvable by standard commercial solvers.

If the recourse function is convex in $K$, then we can compute a subgradient of the epigraph of the recourse function given any specific $K_{t} \in \mathbb{K}$. This subgradient would be a valid inequality, and our decomposition algorithm becomes Benders decomposition algorithm. However, due to the neurons with positive weights, cuts derived by such a method may cut off part of the epigraph. So, the computed solutions may not be globally optimal. In other words, we are “aggressive” in updating $K_{t}$. In this paper, we make use of this aggressiveness and design a multi-cut method to solve Problem (16). At the beginning of the algorithm, we speed up the algorithm via these aggressive cuts; they are not valid but they ensure a fast convergence to a local optimum. After a certain number of iterations, we relax all of the aggressive cuts to valid cuts and then use integer optimality cuts to ensure that the algorithm converges to the global optimum.

Remark 3. Since evaluation of the recourse function is cheap, Problem (16) could be solved by the integer L-shaped algorithm (Laporte and Louveaux 1993). When the first-stage decision $K_{t}^m$ is solved in the $(m - 1)$th iteration, an integer optimality cut is derived and added to the first-stage problem. The integer optimality cut recovers the recourse corresponding to $K_{t}^m$ if the first-stage decision is equal to $K_{t}^m$, and it recovers a lower bound otherwise (see (Laporte and Louveaux 1993) for the convergence proof of this algorithm). However, the integer optimality cut may result in slow convergence since it does not make use of any of the gradient information about the recourse function.
5.2. Aggressive Cuts for the Action Selection Problem

In this subsection, we construct the aforementioned aggressive cuts by making use of the gradient information. Suppose we have solved the first-stage problem in the \((m-1)\)th iteration and computed the first-stage decision \(K^m_t\). In Problem (16), the gradient of the output of neuron \(j \in J\) at point \(K^m_t\) can be computed directly from the definition of ReLU: for all \(n \in \mathcal{N}, j \in J, s \in S_2\), we define

\[
\phi_{n_j}^{ms} \triangleq \begin{cases} 
-w_{j(t+1)}u_{n_j(t+1)}, & \text{if } u_j^T(K^m_t, d^s_{t+1}) + u_{0j(t+1)} > 0, \\
0, & \text{otherwise.}
\end{cases}
\]

We further define

\[
\phi_{0j}^{ms} \triangleq \begin{cases} 
-w_{j(t+1)} \left( \sum_{t'=N+1}^{N+1} u_{j(t+1)}d^s_{t'(t+1)} + u_{0j(t+1)} \right), & \text{if } u_j^T(K^m_t, d^s_{t+1}) + u_{0j(t+1)} > 0, \\
0, & \text{otherwise.}
\end{cases}
\]

Note that \(\phi_{n_j}^{ms}\) and \(\phi_{0j}^{ms}\) depend on \(s \in S_2\). Given \(\phi_{n_j}^{ms}\) and \(\phi_{0j}^{ms}\), we can derive the following cuts for neurons with positive/negative weights, respectively:

\[
\eta^+ \geq \sum_{s \in S_2} \sum_{j \in J^+_1} \left( \sum_{n \in \mathcal{N}} \phi_{n_j}^{ms}_{Knt} + \phi_{0j}^{ms} \right), \quad \forall K_t \in \mathbb{K}, \quad (18)
\]

\[
\eta^- \geq \sum_{s \in S_2} \sum_{j \in J^-_1} \left( \sum_{n \in \mathcal{N}} \phi_{n_j}^{ms}_{Knt} + \phi_{0j}^{ms} \right), \quad \forall K_t \in \mathbb{K}. \quad (19)
\]

**Proposition 4.** Suppose cuts are generated according to (18)–(19):

(i) Cut (19) is a valid inequality for \(epi^-\);

(ii) Cut (18) is not valid for \(epi^+\).

**Proof.** We first prove that Eq. (19) is a valid inequality for \(epi^-\). For all \(j \in J^-_1\), we have \(w_j(t+1) \leq 0\). For all \(s \in S_2\), denote \(\eta^-_j = -w_{j(t+1)} \max \left\{ u_{j(t+1)}^T(K_t, d^s_{t+1}) + u_{0j(t+1)}, 0 \right\} \). We then have

\[
\eta^-_j = \max \left\{ -w_j(t+1) \sum_{n'=1}^{N} u_{n'j(t+1)}K_{n't} - w_j(t+1) \left( \sum_{t'=N+1}^{N+1} u_{j(t+1)}d^s_{t'(t+1)} + u_{0j(t+1)} \right), 0 \right\}
\]

\[
\geq \max \left\{ \sum_{n \in \mathcal{N}} \phi_{n_j}^{ms}K_{nt} + \phi_{0j}^{ms}, 0 \right\}
\]

\[
\geq \sum_{n \in \mathcal{N}} \phi_{n_j}^{ms}K_{nt} + \phi_{0j}^{ms}, \quad \forall K_t \in \mathbb{K}.
\]
The first equality is true because $w_{j(t+1)} \leq 0$, so the max operator is unchanged. The second line follows from the definitions of $\phi_{nj}^{ms}$ and $\phi_{0j}^{ms}$. If $u^T_{j(t+1)} (K_t, d_{t+1}^s) + u_{0j(t+1)} > 0$, the equality holds trivially; if $u^T_{j(t+1)} (K_t, d_{t+1}^s) + u_{0j(t+1)} \leq 0$, then $\phi_{nj}^{ms}$ and $\phi_{0j}^{ms}$ are all zero and so the inequality holds because the second line is always non-negative. Summing up over all neurons $j \in J_{t+1}$ and then taking expectations on both sides of the inequality, we have

$$\eta^- \geq \sum_{s \in S_2} \sum_{j \in J_{t+1}^-} \eta^*_{j}$$

$$\geq \sum_{s \in S_2} \sum_{j \in J_{t+1}^-} \left( \sum_{n \in N} \phi_{nj}^{ms} K_{nt} + \phi_{0j}^{ms} \right), \quad \forall K_t \in \mathbb{K}.$$ 

Thus, according to the definition of $\text{epi}^-_{t}$, we have

$$\text{epi}^-_{t} \subset \left\{ (K_t, \eta) \in \mathbb{R}^{N+1} \left| \eta^- \geq \sum_{s \in S_2} \sum_{j \in J_{t+1}^-} \left( \sum_{n \in N} \phi_{nj}^{ms} K_{nt} + \phi_{0j}^{ms} \right) \right. \right\},$$

and so Eq. (19) is a valid inequality.

To show that Eq. (18) is not a valid inequality for $\text{epi}^+_{t}$, we only need to find a counterexample. Consider a simple case with only one facility and $\mathbb{K} = \{ K_t \in \mathbb{Z} | 0 \leq K_t \leq 2 \}$. Suppose the recourse function for a node $j \in J_{t+1}$ is $v_{t+1} = -\max\{ K_t - 1, 0 \}$. Then, for $K_t^m = 0$, we can easily verify $\phi_{nj}^{ms} = \phi_{0j}^{ms} = 0$. However, for $K_t = 2$, we have $\eta^*_j = -1 \times \max\{ 2 - 1, 0 \} < 0$, and so the inequality $\eta^*_j \geq \sum_{n \in N} \phi_{nj}^{ms} K_{nt} + \phi_{0j}^{ms}$ is not satisfied for all $K_t \in \mathbb{K}$. □

Since Cut (18) is not valid, it may cut off parts of the epigraph and overestimate the costs generated along this direction. However, we can use this cut to achieve faster convergence, but we need to later transform it into valid inequality in order to derive the global optimum.

**Remark 4.** The aggressive cuts, i.e. Cuts (18)–(19), are constructed based on the gradients of the epigraph of the recourse function at given first-stage decisions $K^m_t$ in the $(m-1)$th iteration. Therefore, if we update the first-stage decisions according to the gradient information, the algorithm can generally lead to a descent direction for the objective function. Compared to those cuts that do not use any gradient information of the recourse function, e.g. the integer optimality cuts in (Laporte and Louveaux 1993), our aggressive cuts can generally lead to a quicker descent in the objective value in each iteration.
5.3. Valid Inequalities for the Action Selection Problem

To derive a valid cut for neurons $j \in J_{t+1}^+$, we need to find a supporting hyperplane for the non-convex set $\text{epi}_{t+1}$. Let $\{\hat{K}_1, \ldots, \hat{K}_L\}$ denote the vertices of the smallest rectangle that contains $K$, where each $\hat{K}_i \triangleq (\hat{K}_{i1}, \ldots, \hat{K}_{iN})$ is an $N$-dimensional vector. Denote

$$
\psi_{jt}(K, d) \triangleq \left\{ u_{jt}^T (K, d) + u_{0jt}, 0 \right\}, \quad \forall (K, d) \in S, j \in J, t \in T \setminus \{T\},
$$

as the output of neuron $j$. We further denote

$$
X \triangleq \begin{bmatrix}
1 & \hat{K}_{11} & \cdots & \hat{K}_{N1} \\
1 & \hat{K}_{12} & \cdots & \hat{K}_{N2} \\
\vdots & \vdots & \ddots & \vdots \\
1 & \hat{K}_{1L} & \cdots & \hat{K}_{NL}
\end{bmatrix} \quad \text{and} \quad Y_j^s \triangleq \begin{bmatrix}
\psi_{jt+1}(\hat{K}_{11}, d_{t+1}) \\
\vdots \\
\psi_{jt+1}(\hat{K}_{L1}, d_{t+1})
\end{bmatrix}, \quad \forall s \in S_2,
$$

as a $(1 + N) \times L$ matrix and an $L$ column vector respectively. For all $s \in S_2, j \in J_{t+1}^+$, define

$$
\theta_{nj}^s \triangleq \left[ (X^T X)^{-1} X^T Y_j^s \right]_n, \quad \forall n \in \mathcal{N}, \quad (20)
$$

and

$$
\theta_{0j}^s \triangleq \max_{l \in \{1, \ldots, L\}} \left\{ \psi_{jt+1}(\hat{K}_{l}, d_{t+1}) - \sum_{n \in \mathcal{N}} \theta_{nj}^s \hat{K}_{nt} \right\}.
$$

Essentially, the above procedure for calculating Eq. (20) finds a hyperplane for some labeled points $(\hat{K}_l, \psi_{jt+1}(\hat{K}_l, d_{t+1}))_{l=1,\ldots,L}$ such that the distance of the points to the hyperplane is minimized. Essentially, it is doing linear regression on these points. Then, the resulting hyperplane can be shifted such that it is larger than $\psi_{jt+1}(\hat{K}_l, d_{t+1})$ for all $l \in \{1, \ldots, L\}$. Thus, we obtain a supporting hyperplane for the labeled samples. In fact, the only requirement for the above result to hold is $\theta_{nj}^s < \infty$ for all $n \in \mathcal{N}$.

**Proposition 5.**

$$
\eta^+ \geq \sum_{s \in S_2} \sum_{j \in J_{t+1}^+} \left[ -w_{jt+1} \left( \sum_{n \in \mathcal{N}} \theta_{nj}^s K_{nt} + \theta_{0j}^s \right) \right], \quad \forall K_t \in \mathbb{K}, \quad (21)
$$

is a valid inequality for $\text{epi}_{t+1}$. 
Proof. It is easy to verify that $\theta_{n_j}^s < \infty$ for all $n \in \mathcal{N}$, as the entries of $X$ and $Y$ are bounded.

Then, given any $s \in \mathcal{S}_2, j \in \mathcal{J}_{t+1}^+$, $t \in \mathcal{T}$, we have

$$
(\theta_j^s) \K_t + \theta_{0j}^s = (\theta_j^s) \K_t + \max_{l \in \{1, \ldots, L\}} \left\{ \psi_{j(t+1)} (\hat{K}_{lt}, d_{t+1}^s) - (\theta_j^s) \K_t \right\}
$$

$$
= \max_{l \in \{1, \ldots, L\}} \left\{ \psi_{j(t+1)} (\hat{K}_{lt}, d_{t+1}^s) + (\theta_j^s) (K_t - \hat{K}_t) \right\}.
$$

The first equation holds by the definition of $\theta_{0j}^s$, and the second one holds since $(\theta_j^s)^\top K_t$ is independent of $l \in \{1, \ldots, L\}$. We introduce

$$
g_j^s (\hat{K}_t) \triangleq \psi_{j(t+1)} (\hat{K}_{lt}, d_{t+1}^s) + \theta_j^s (K_t - \hat{K}_t), \quad \forall l \in \{1, \ldots, L\},
$$

to simplify the notation. Then, for all $l_1, \ldots, l_4 \in \{1, \ldots, L\}$ and $\tau_1, \ldots, \tau_3 \in [0, 1]$, we have

$$
(\theta_j^s) \K_t + \theta_{0j}^s = \max_{l \in \{1, \ldots, L\}} \left\{ g_j^s (\hat{K}_t) \right\}
$$

$$
\geq \tau_3 \left( \tau_1 g_j^s (\hat{K}_{l_1}) + (1 - \tau_1) g_j^s (\hat{K}_{l_2}) \right) + (1 - \tau_3) \left( \tau_2 g_j^s (\hat{K}_{l_3}) + (1 - \tau_2) g_j^s (\hat{K}_{l_4}) \right)
$$

$$
\geq \tau_3 g_j^s (\tau_1 \hat{K}_{l_1} + (1 - \tau_1) \hat{K}_{l_2}) + (1 - \tau_3) \left[ \tau_2 \hat{K}_{l_3} + (1 - \tau_2) \hat{K}_{l_4} \right]
$$

$$
\geq g_j^s \left[ \tau_3 [\tau_1 \hat{K}_{l_1} + (1 - \tau_1) \hat{K}_{l_2}] + (1 - \tau_3) \left[ \tau_2 \hat{K}_{l_3} + (1 - \tau_2) \hat{K}_{l_4} \right] \right].
$$

The second line holds because the largest point of a set is larger than the convex combination of any two points within the set. The third and forth lines hold because $\psi_j (\cdot, d_{t+1}^s)$ is convex for any $d_{t+1}^s$. As a consequence of the above inequality, we have

$$
(\theta_j^s)^\top K_t + \theta_{0j}^s \geq \psi_j (K_t, d_{t+1}^s), \quad \forall K_t \in \mathcal{K}.
$$

This inequality follows for any $K_t \in \mathcal{K}$, since there exists $l_1, \ldots, l_4 \in \{1, \ldots, L\}$ and $\tau_1, \ldots, \tau_3 \in [0, 1]$ such that

$$
K_t = \tau_3 \left[ \tau_1 \hat{K}_{l_1} + (1 - \tau_1) \hat{K}_{l_2} \right] + (1 - \tau_3) \left[ \tau_2 \hat{K}_{l_3} + (1 - \tau_2) \hat{K}_{l_4} \right].
$$

Since $w_{jt} > 0$ for all $j \in \mathcal{J}_{t+1}^+$, we have

$$
- w_{j(t+1)} \left( \sum_{n \in \mathcal{N}} \theta_{n_j}^s n_t + \theta_{0j}^s \right) \leq - w_{j(t+1)} \psi_{j(t+1)} (K_t, d_{t+1}^s), \quad \forall K_t \in \mathcal{K}, s \in \mathcal{S}_2.
$$
Summing up the above inequalities over \( j \in J_{t+1}^+ \) and taking expectations on both sides yields

\[
- \sum_{s \in S_2} \sum_{j \in J_{t+1}^+} w_{jt} \left( \sum_{n \in N} \theta_{nj}^s K_{nt} + \theta_{0j}^s \right) \leq - \sum_{s \in S_2} \sum_{j \in J_{t+1}^+} w_{jt} \psi_j(t+1) \left( K_t, d_t^s \right), \quad \forall K_t \in K.
\]

It can be seen that the cut provided by Eq. (21) is a lower bound of \( epi_t^+ \), and thus it is valid. \( \square \)

Valid inequalities for neurons \( j \in J_{t+1}^+ \) can also be derived by constructing the convex hull of \( epi_t^+ \), but then we would have to solve a linear programming problem to calculate the convex hull for each neuron \( j \in J_{t+1}^+ \) and each realization \( s \in S_2 \) (Sen and Sherali 2006). Given that \( |J_{t+1}^+| \) and \( |S_2| \) can be large, this method would be computationally expensive. In contrast, Cut (21) can be computed with basic matrix operations—i.e. inverse, multiplication, maximum, et cetera—which are all much cheaper. Though the computation of the inverse of \( X \) in \( \theta_{nj}^s \) can be expensive (the number of terms \( \hat{K}_t \) increases exponentially with the number of facilities), this method is still workable for a typical case with ten to fifteen facilities. In addition, Cut (21) is independent of the number of iterations, so we only need to solve it once for a specific state sample in \( S_1 \).

### 5.4. A Mixed-Cuts Decomposition Algorithm

We now present a mixed-cuts decomposition (MCD) algorithm for solving Problem (16). The core idea of this algorithm originates in Benders decomposition, but we use aggressive cuts in the beginning and then later relax all of the aggressive cuts to Cuts (21). Cuts (21) are valid, but they may not be tight. Their purpose is to ensure that the algorithm converges to the global optimum.

To achieve global optimality, we introduce the integer optimality cuts from (Laporte and Louveaux 1993) into our algorithm. Define \( \zeta^m (K_t) \) to be a function in the \( m \)th iteration, where \( \zeta^m (K_t) = 0 \) if \( K_t = K_t^m \) and \( \zeta^m (K_t) \geq 1 \) otherwise (to formulate the indicator function, we may need to transform the general integer variables \( K_t \) to binary variables). Denote \( \eta \) as a lower bound for the recourse function. The integer optimality cuts can then be derived from (Laporte and Louveaux 1993):

\[
\eta \geq v_{t+1} (K_t^m, d_t) - \zeta^m (K_t) \left[ v_{t+1} (K_t^m, d_t) - \eta \right],
\]

where \( v_{t+1} (\cdot) \) is provided by Eq. (17). We see that this cut recovers \( v_{t+1} (K_t^m, d_t) \) if \( \zeta^m (K_t) = 0 \), i.e. \( K_t = K_t^m \), and recovers a lower bound otherwise.
With these cuts, the first-stage problem in the $m$th iteration of the MCD algorithm can be approximated with

$$
\min \ c_t \ (K_{t-1}, K_t) + \gamma \eta \quad (FP)
$$

$$
s.t. \ \eta \geq v_{t+1} \left( K_{t+1}^m - \zeta_{m'}^t (K_t) \left[ v_{t+1} (K_{t+1}^m, d_t) - \eta \right] \right), \ \forall m' \in C_1^m,
$$

$$
\eta \geq \sum_{s \in S_2} \sum_{j \in J_t} \left( \sum_{n \in N} \phi_{n_j}^{m'} K_{nt} + \phi_{0_j}^{m'} \right) - w_0(t+1), \ \forall m' \in C_2^m,
$$

$$
\eta \geq \sum_{s \in S_2} \left[ \sum_{j \in J_t^+} \left( \sum_{n \in N} \phi_{n_j}^{m'} K_{nt} + \phi_{0_j}^{m'} \right) - \sum_{j \in J_t^+} w_j(t+1) \left( \sum_{n \in N} \theta_{n_j}^m K_{nt} + \theta_{0_j}^m \right) \right] - w_0(t+1), \ \forall m' \in C_3^m,
$$

$$
K_t \in \mathbb{K},
$$

where $C_1^m$, $C_2^m$ and $C_3^m$ are the sets of indices in the $m$th iteration for Cuts (22), (23), and (24), respectively. In the above problem, Cuts (22) are the integer optimality cuts generated up to iteration $m$. Cuts (23) are the aggressive cuts from combining Eqs. (18) and (19). Note that these cuts are not valid since Eq. (18) is not valid for $\text{epi}_t^+ (d_t)$. Cuts (24) are valid inequalities when Eq. (18) is relaxed to Eq. (21) for neurons $j \in J_t^+$. 

Our algorithm has two phases. In Phase 1, Cuts (22)–(23) are simultaneously added to Problem $(FP)$ to update the first-stage decisions. Then, we relax Cuts (23) to (24) in each iteration, and stop adding aggressive cuts when the algorithm reaches a preset number of iterations. In Phase 2, only Cuts (22), i.e. the integer optimality cuts, are added into Problem $(FP)$ to achieve the global optimum.

**Phase 1: adding multiple cuts for Problem $(FP)$** In the $m$th iteration, we solve Problem $(FP)$ to obtain its optimal solution $K_{t}^{m+1}$. Then, we have the following two steps.

1. Set $\bar{C}_1^m = C_1^m \cup \{m + 1\}$ and $\bar{C}_2^m = C_2^m \cup \{m + 1\} \cap \{1, \ldots, m_1\}$ if $K_{t}^{m+1} \notin \left\{ K_{t}^{m'} \right\}_{m' \in C_1^m}$; that is, if the updated $K_{t}^{m+1}$ does not appear before, we include the corresponding Cuts (22) and (23) in the next iteration.
2. Select one of Cuts (23) up to iteration $m$ and relax it to Cut (24). The criterion for this choice may be the minimum slackness:

$$m_0 = \arg \min_{m' \in \mathcal{C}^m_2} \left( r^{m+1} - \sum_{s \in S} \sum_{j \in J} \left( \sum_{n \in N} \phi^{m'}_{nj} K^{m+1} + \phi^{m'}_{n0} \right) \right).$$

Then, we set $\mathcal{C}^{m+1}_2 = \mathcal{C}^m \setminus \{m_0\}$ and $\mathcal{C}^{m+1}_3 = \mathcal{C}^m_3 \cup \{m_0\}$, and move to the next iteration.

The first step is to ensure that Problem (FP) does not get stuck with the same decisions in Phase 1, as $K_t$ are integer variables and may be unchanged if we just relax the inequality with the minimum slackness.

**Example 1.** Consider a simple optimization problem with the objective function being max $x + y$ and the constraints $x, y \in \{0, 1\}$, $x \leq 1$ and $-x + y + 1.5 \geq 0$. One of the optimal solution is $(x^*, y^*) = (1, 0)$ and the constraint with the minimum slackness is $x \leq 1$. If we relax $x \leq 1$ to $x \leq 2$, the optimal solution is unchanged, but if we relax $-x + y + 1.5 \geq 0$ to $-x + y + 2 \geq 0$, the optimal solution becomes $(x^*, y^*) = (1, 1)$.

Also, we set a preset $\bar{m}_1$ such that we stop adding Cuts (23) but keep relaxing Cuts (23) to Cuts (24), which are valid, when $m \geq \bar{m}_1$. Therefore, the corresponding index set $\mathcal{C}^m_2$ will become empty after a finite number of iterations.

**Phase 2: adding integer optimality cut for Problem (FP)** If all of the existing Cuts (23) have been relaxed—i.e. $\mathcal{C}^m_2$ is empty—then we stop the first phase and enter the second phase. In the second phase, we set $\mathcal{C}^{m+1}_1 = \mathcal{C}^m_1 \cup \{m + 1\}$, $\mathcal{C}^{m+1}_2 = \mathcal{C}^m_2$, and $\mathcal{C}^{m+1}_3 = \mathcal{C}^m_3$, so that only the integer optimality cuts (Cuts (22)) are added in each iteration. This phase is similar to the integer L-shape algorithm except for the additional valid cuts generated in the first phase.

**Stopping criterion** A lower bound can be obtained by solving Problem (FP) when $\mathcal{C}^m_2 = \emptyset$. An upper bound can be obtained from the objective value of the best-found decision up to iteration $m$. Therefore, one can terminate the algorithm when the difference between the upper and lower bounds falls below a preset precision. The finite convergence of the MCD algorithm can be established as follows.
Theorem 3. The MCD algorithm yields a globally optimal solution in a finite number of iterations if Phase 1 has a finite number of iterations.

Proof. If Phase 1 ends in a finite number of steps, we have $C_2^{m_2} = \emptyset$ and there are no Cuts (23) in Problem ($FP$). According to [Laporte and Louveaux 1993, Proposition 2] and Proposition 5, Cuts (22) and (24) are all valid. Since the action space $K$ is finite, Problem ($FP$) has finitely many feasible solutions. As a result, in Phase 2, there are only finitely many Cuts (22) that can be added in Problem ($FP$). Thus, the algorithm can converge to the global optimal solution in a finite number of iterations. □

According to Proposition 3, the algorithm can converge to the global optimum if $m_1$ is finite. We also set a maximum iteration count $\bar{m}_2$ for the algorithm so that it terminates before the gap between the upper and lower bounds reaches the preset precision. After Phase 1 ends, the convergence to the global optimum is ensured by Cuts (22), which can be slow when the problem size is large. However, as we will see in our numerical study, the MCD algorithm finds a high-performance solution in a relatively small number of iterations. Therefore, one can choose a suitable $\bar{m}_2$ to trade-off between the performance of the algorithm and its CPU time. The flow of the NN-FVI algorithm combined with MCD is summarized in Figure 3, and the detailed procedure of MCD is presented in Algorithm 2.

6. Numerical Studies

We test the performance of our proposed method in this section, in three parts. First, we compare the performance of the ReLU-based NN-FVI to those with other types of activation functions. We verify that ReLU outperforms others in solving MCIP. Second, we compare the performance of the proposed MCD algorithm with the exhaustive enumeration method (brute-force method) in the action selection problem. Third, we combine NN-FVI with MCD and test its performance in a case study where we analyze its economic performance over an inflexible counterpart. The inflexible counterpart, which has no capacity adjustment options, is modeled as a two-stage capacity investment problem and solved with Benders decomposition. These numerical studies are performed
Start

$t = T$

**Step 1:** Generate $S_1$ samples from the state space

**Step 2:** For each sample $s \in S_1$, find the optimal actions via the MCD algorithm and calculate $\hat{V}_t(K_{t-1}^s, d_t^s)$

$t = t - 1$

$t > 1$

Yes

No

**Step 3:** Given the labeled samples, train the neural network $\hat{V}_t(\cdot; u_t, w_t)$

Given the initial state $(0_u, a_0)$, find the optimal action via the MCD algorithm

End

Figure 3  Flowchart of the NN-FVI algorithm combined with MCD

on a workstation with an Intel Xeon E5-2665 processor and 32 GB RAM in the Matlab R2016b environment. The neural networks are trained by the Levenberg–Marquardt algorithm via the neural network toolbox of Matlab (Beale et al. 2018).

6.1. ReLU Outperforms Other Activation Functions in Solving MCIP

In this subsection, we test the performance of NN-FVI with different types of activation functions, including ReLU, tanH (hyperbolic function), and sigmoid. Here, the action selection problem is solved by the brute-force method as MCD is not applicable to the networks using tanH and sigmoid functions.

In Table 2, we test a small-scale case (Case 1.1) with discrete demands that it is solvable by DP, and compare the approximated objective values derived from NN-FVI to the exact objective values derived from DP; thus, the closer the value to the benchmark, the better the approximation is. As can be seen, the exact ENPV derived from DP is 357.9, and the approximate objective derived from NN-FVI with ReLU is 357.7. However, the approximated ENPVs derived from NN-FVI with tanH and sigmoid functions are 441.3 and 628.9 respectively, both of which are far from the exact value.
Algorithm 2 The MCD algorithm

\textbf{Input:} $\bar{m}_1, \bar{m}_2, \epsilon$, state $(K_{t-1}, d_t)$, initial solution $K_t^1$

\textbf{Output:} $K_t^*$

1: Initialize $m = 1$, $V_{lb} = -\infty$, $V_{ub} = +\infty$, $C_{1}^m = \{1\}$, $C_{2}^m = \{1\}$, $C_{3}^m = \emptyset$

2: \textbf{while} $m \leq \bar{m}_1$ and $C_{2}^m \neq \emptyset$ \textbf{do}

3: \hspace{1em} Evaluate $V^m = c_t(K_{t-1}, K_t^m) + \gamma v_{t+1}(K_t^m, d_t)$ and set $V_{ub} = \min\{V^m, V_{ub}\}$

4: \hspace{1em} Given $K_t^m$, construct Cuts (22) & (23)

5: \hspace{1em} \textbf{if} $K_t^{m+1} \notin \{K_t^m\}_{m' \in C_{1}^m}$ \textbf{then}

6: \hspace{2em} $C_{1}^{m+1} = C_{1}^m \cup \{m + 1\}$ and $C_{2}^{m+1} = C_{2}^m \cap \{1, \ldots, \bar{m}_1\}$

7: \hspace{1em} \textbf{end if}

8: \hspace{1em} Compute $m_0 = \arg\min_{m' \in C_{2}^m} \left( \eta^{m+1} - \sum_{s \in S_2} \sum_{j \in S} (\sum_{n \in \mathcal{N}} y_{n}^{m'} K_t^{m+1} + \phi_{0j}^{m'}) \right)$

9: \hspace{1em} Given $K_t^{m_0}$, construct Cut (24)

10: \hspace{1em} Set $C_{2}^{m+1} = \tilde{C}_{2}^m \setminus \{m_0\}$ and $C_{3}^{m+1} = C_{3}^m \cup \{m_0\}$

11: \hspace{1em} $m \leftarrow m + 1$

12: \textbf{end while}

13: \textbf{while} $m < \bar{m}_2$ and $V_{ub} - V_{lb} < \epsilon$ \textbf{do}

14: \hspace{1em} Evaluate $V^m = c_t(K_{t-1}, K_t^m) + \gamma v_{t+1}(K_t^m, d_t)$ and set $V_{ub} = \min\{V^m, V_{ub}\}$

15: \hspace{1em} Given $K_t^m$, construct Cut (22)

16: \hspace{1em} Solve Problem (FP) and derive $K_t^{m+1}$ and $V_{lb}$

17: \hspace{1em} Set $C_{1}^{m+1} = C_{1}^m \cup \{m + 1\}$, $C_{2}^{m+1} = C_{2}^m$, and $C_{3}^{m+1} = C_{3}^m$

18: \hspace{1em} $m \leftarrow m + 1$

19: \textbf{end while}

20: $K_t^* = \arg\min_{K \in \{K_1, \ldots, K_T\}} \{c_t(K_{t-1}, K) + \gamma v_{t+1}(K, d_t)\}$

For a case with a larger size, DP is not applicable due to the curse of dimensionality. Instead, an inflexible two-stage stochastic capacity investment model is selected as the benchmark. We perform out-of-sample tests on the optimal policies derived from both the inflexible method and NN-FVI, on an identical sample set with 10,000 sample paths; in this case, a better policy should
Table 2 Comparisons of NN-FVI with different activation functions for small-scale cases

| Algorithms  | Activation fun. | CPU time | Obj. values* | Relative gaps |
|-------------|----------------|----------|--------------|---------------|
| Case 1.1    |                |          |              |               |
| (I, N, T)   |                |          |              |               |
| 2, 2, 2     |                |          |              |               |
| DP          | -              | 160 s    | 357.9        | -             |
| NN-FVI      | ReLU           | 6 s      | 357.7        | <0.1%         |
| NN-FVI      | tanH           | 10 s     | 441.3        | 23.3%         |
| NN-FVI      | Sigmoid        | 10 s     | 628.9        | 75.7%         |

* The ENPVs are derived from the approximated objective values directly.

Table 3 Comparisons of NN-FVI with different activation functions for medium-scale cases

| Algorithms  | Activation fun. | CPU time | ENPV  | VoF  |
|-------------|----------------|----------|-------|------|
| Inflexible design |                | 306 s    | 1530.9| -    |
| Case 1.2    |                |          |       |      |
| (I, N, T)   |                |          |       |      |
| 4, 3, 10    |                |          |       |      |
| NN-FVI      | ReLU           | 21 040 s | 1634.9| 104.0|
| NN-FVI      | tanH           | 25 624 s | 1625.4| 94.5 |
| NN-FVI      | Sigmoid        | 27 452 s | 1518.2| -12.7|

derive a higher ENPV in the out-of-sample tests. In Table 3, simulation results of a medium-scale case with (I, N, T) = (4, 3, 10) indicate that the optimal policy derived from NN-FVI with ReLU outperforms the policies derived from networks with tanH and sigmoid functions. Also, the CPU time of NN-FVI with ReLU is much less than those of the NN-FVI with tanH and sigmoid, as the piecewise linear activation functions are easier to use in calculation compared to the nonlinear ones. We can conclude that the neural network with ReLU is a better approximator in comparison to those using tanH and sigmoid.

6.2. The Action Selection Procedure Can be Solved in Reasonable Time

In this subsection, we compare the performance of the MCD algorithm with two other alternatives—(1) the brute-force algorithm and (2) the integer L-shaped algorithm. In the integer L-shaped algorithm, only Cuts (22) (integer optimality cuts) are added in each iteration.

Four neural networks with different sizes, i.e. Cases 2.1–2.4, are randomly generated, and each is formulated as Problem (16) and solved by the aforementioned algorithms. The performance of the algorithms is measured in terms of the CPU time and their best-found objective value achieved before the algorithm is terminated. As can be seen, in Cases 2.1 and 2.2, the CPU time of the brute-force method is around 11.7 seconds. For the integer L-shaped algorithm, it takes around 20 seconds to converge to the optimal solution, which is slower than the brute-force method. For
### Table 4  Simulation results of the MCD algorithm

| Algorithms     | Stop criterion* | Iterations | CPU time | Exp. profits | Relative gaps |
|----------------|-----------------|------------|----------|--------------|---------------|
| Case 2.1       | Brute-force      | -          | -        | 11.7 s       | 2506.0        | -             |
| (N = 3)        | Integer L-shaped| 5%/200 steps| 178      | 20.6 s       | 2506.0        | 0%            |
| MCD            | 5%/200 steps     | 112        | 13.1 s   | 2506.0        | 0%            |
| Case 2.2       | Brute-force      | -          | -        | 11.8 s       | 6731.9        | -             |
| (N = 3)        | Integer L-shaped| 5%/200 steps| 189      | 22.6 s       | 6731.9        | 0%            |
| MCD            | 5%/200 steps     | 95         | 10.1 s   | 6731.9        | 0%            |
| Case 2.3       | Brute-force      | -          | -        | 53.1 s       | 119.3 × 10^6  | -             |
| (N = 4)        | Integer L-shaped| 5%/200 steps| 200      | 26.5 s       | 79.8 × 10^6   | 33.11%        |
| MCD            | 5%/100 steps     | 100        | 18.9 s   | 119.2 × 10^6  | <0.1%         |
| MCD            | 5%/200 steps     | 200        | 30.0 s   | 119.2 × 10^6  | <0.1%         |
| Case 2.4       | Brute-force      | -          | -        | 709.7 s      | 256.8 × 10^6  | -             |
| (N = 5)        | Integer L-shaped| 5%/200 steps| 200      | 66.2 s       | 233.1 × 10^6  | 9.23%         |
| MCD            | 5%/100 steps     | 100        | 15.6 s   | 256.7 × 10^6  | <0.1%         |
| MCD            | 5%/200 steps     | 200        | 47.0 s   | 256.8 × 10^6  | 0%            |

*The algorithm is stopped if the relative gap is smaller than 5% or the total number of iteration reaches 200.

MCD, the CPU time for these two cases decreases to 13.1 and 10.1 seconds, respectively, which is close to the brute-force method. This is not surprising since the brute-force method is efficient when the problem size is small. By increasing the number of facilities from three to five, we can see from Table 4 that the CPU time of the brute-force method increases to 709 seconds. On the contrary, MCD achieves the global optimum within 47 seconds. Based on this evidence, we speculate that the CPU time of the MCD algorithm might be much less than the brute-force method in even larger scale cases.

We now analyze the convergence of MCD and the integer L-shaped algorithm. As can be seen, the MCD algorithm converges to the global optimum in fewer iterations than the integer L-shaped algorithm. In Cases 2.1 and 2.2, where the number of facilities is $N = 3$, the integer L-shaped method can still converge to the global optimum within 200 iterations. However, when the number of facilities is increased, the best-found solutions by the integer L-shaped method within 200 iterations are sub-optimal. In particular, the relative gaps of the best-found objective values and the global optimums for cases $N = 4$ and $N = 5$ are 33.11% and 9.23%, respectively. In contrast, MCD can find the global optimum (or a near-optimal solution) in around 100 steps. These numerical results verify Remark 4, where we suggest that MCD ensures faster convergence by using the aggressive cuts.
6.3. Verify the Performance of NN-FVI in a Real Case

This case study on a multi-facility waste-to-energy (WTE) system is adapted from Ref. (Zhao et al. 2018). In the reference, the system can only expand the capacity of the facilities, while in this paper, capacity contraction is allowed. In addition, we adjust the original case into a smaller one where we can perform practical sensitivity analysis. The WTE system has four candidate sites located in different sectors. The facilities at each site are able to dispose food waste collected from each sector by using an anaerobic digestion technique, which transforms the food waste into electricity. Undisposed waste will be subjected to further treatment via landfill, incurring greater disposal costs, i.e. penalties. The revenue comes from selling the electricity and the salvage values of the contracted capacity, and the costs consist of the disposal, penalty, and transportation costs.
as well as the capacity expansion costs. The data and parameters of the case study are found in (Zhao et al. 2018).

Sensitivity analysis is implemented on the ratio of the per unit salvage value with the per unit expansion cost (S/E ratio), i.e. $q_{nt}^-/q_{nt}^+$. If the S/E ratio is one, it means that the salvage value for per unit capacity is equivalent to the per unit expansion cost. As can be seen in Table 5, the percentage improvement of the system with a flexible design over an inflexible design decreases as the S/E ratio increases. When gamma is 0.862 and the S/E ratio is 0, the expected net present value (ENPV) of the inflexible design is 23.9 million, while the ENPV of the flexible design is 35.0 million. In this case, the system performance is improved by 46.4% when the capacity is flexible. However, this improvement decreases to 15.1% when the S/E ratio increases to 0.99. On the other hand, we see that if the discount factor $\gamma$ is close to one, the improvement may become negative: the VoF becomes $-0.6$ when both $\gamma$ and the S/E ratio are 0.99. This is reasonable since the decision maker can establish facilities with large capacity in the beginning and then salvage all of them in the last period, without suffering a significant loss. In other words, the two-stage model may yield an optimal solution for the problem when both $\gamma$ and the S/E ratio are large. Meanwhile, NN-FVI is based on approximation so it may sometimes underestimate the ENPV.

7. Extensions of the NN-FVI with the MCD Algorithm

We focus on solving MCIP in this paper, but our method is applicable to many other problems where the action space is finite and high dimensional, such as the following.

1. **Lead time.** If we consider the MCIPs where the capacity adjustment has lead time, we can formulate the capacity under construction as part of the state variables. In this case, we may have a two-layer neural network with additional inputs, but the action selection problem is not affected.

2. **Uncertain rewards/costs.** The proposed method can also solve MCIPs with uncertain parameters. For example, if the rewards of the system are uncertain, we can model the rewards as state variables.
Table 5  Sensitivity analysis given different S/E ratio and gamma

| Gamma (γ) | S/E ratio ($q^-_{nt}/q^+_{nt}$) | ENPV ($\times 10^6$ S$)$ | VoF ($\times 10^6$ S$)$ | Improvement (%) |
|-----------|---------------------------------|--------------------------|--------------------------|-----------------|
| Inflexible design | NN-FVI | |
| 0.862 | 0 | 23.9 | 35.0 | 11.1 | 46.4 |
| | 0.25 | 33.7 | 45.2 | 11.5 | 34.1 |
| | 0.50 | 43.8 | 55.6 | 11.8 | 26.9 |
| | 0.75 | 54.3 | 65.7 | 11.4 | 21.0 |
| | 0.99 | 64.9 | 74.7 | 9.8 | 15.1 |
| 0.923 | 0 | 93.8 | 106.4 | 12.6 | 13.4 |
| | 0.25 | 115.1 | 129.8 | 14.7 | 12.8 |
| | 0.50 | 137.2 | 152.4 | 15.2 | 11.1 |
| | 0.75 | 160.4 | 171.0 | 10.6 | 6.6 |
| | 0.99 | 184.0 | 188.6 | 4.6 | 2.5 |
| 0.99 | 0 | 204.3 | 220.7 | 16.4 | 8.0 |
| | 0.25 | 248.9 | 265.9 | 17.0 | 6.8 |
| | 0.50 | 295.2 | 311.8 | 16.6 | 5.6 |
| | 0.75 | 345.9 | 350.3 | 4.4 | 1.3 |
| | 0.99 | 398.4 | 398.0 | -0.6 | -0.2 |

3. Different types of capacity adjustment costs. If the capacity adjustment costs contain fixed costs or are non-convex, the proposed MCD algorithm can still solve the action selection problem. However, we may have to then solve a mixed integer nonlinear programming problem to update the first-stage decision in each iteration if the costs are non-convex.

Earlier in this paper, we compare the economic performance of NN-FVI with an inflexible two-stage MCIP model via out-of-sample tests. However, as indicated by Zhao et al. (2018), the complexity of the out-of-sample tests for ADP can be even higher than solving the original ADP. To address this, one possible solution to simplify the out-of-sample tests is to approximate the policy of the NN-FVI offline. For example, after solving the action selection problem for each state sample, we can do function fitting on these samples and approximate the policy policy. When the policy is approximated offline, it does not add to the cost of the original NN-FVI algorithm. Then, we can use this approximate policy in the out-of-sample tests. In this case, the out-of-sample tests are more tractable and it is easier to implement the resulting policy in practice. Future work can consider how to approximate the policy to achieve high precision in the out-of-sample tests.
8. Conclusions

This paper studies an MCIP where the capacity of the facilities can be either expanded or contracted in each decision period. To solve this problem, we formulate it as an MDP, and analyze the structure of its value functions. Then, an NN-FVI algorithm is proposed, where the value functions are approximated with a two-layer neural network with ReLU activation functions. The consistency of NN-FVI is also formally proved. Since the action selection procedure of NN-FVI is time-consuming, we formulate it as a two-stage stochastic programming problem with a non-convex recourse function, and we design an MCD algorithm to solve it. We verify that the MCD algorithm converges to the global optimum in a finite number of iterations, and our numerical studies show that it significantly speeds up the action selection problem when compared with the brute-force method and the integer L-shape algorithm.

Though we have verified that the MCD algorithm converges to the global optimum in a finite number of steps, the convergence can still be slow when the problem size is large. One future direction for us is to use other types of valid cuts, or to construct a tight convex hull for the value function, when selecting the optimal actions. Another potential direction is to use multi-layer neural networks with broader applicability. In addition, we may apply the proposed methods in solving more sophisticated capacity investment problems, such as the problems with non-convex capacity expansion costs, to investigate the relationship between the time value of money and the economies of scale.

Appendix. Proof of Theorem 2

This result is derived from (Munos and Szepesvári 2008, Corollary, 4). Since we have proved Proposition 2 and Lemma 2, the only thing we need to prove now is that the MCIP presented in this paper satisfies the MDP regularity assumption and uniformly stochastic transitions assumption. Denote $S$ and $A$ as the state space and the action space of the MDP respectively, where $x \in S$ and $a \in A$. Let $P_{R_t}(R_t \in \mathbb{R}|x,a)$ be the probability of reward $R_t$ given state $x \in S$ and action $a \in A$ in time $t$, and let $R_{\text{max}}$ be a positive real number. The assumptions stated in (Munos and Szepesvári 2008) are presented below.
ASSUMPTION 5. [MDP regularity] The MDP satisfies the following conditions: $S$ is bounded, closed subset of some Euclidean space, $A$ is finite and the discount factor $\gamma$ satisfies $0 < \gamma < 1$. The reward kernel is such that the immediate reward function is a bounded measurable function with bound $R_{\text{max}}$. Further, the support of $P_{\text{R}}(\cdot|x,a)$ is included in $[-R_{\text{max}}, R_{\text{max}}]$ independently of $(x,a) \in S \times A$.

ASSUMPTION 6. [Uniformly stochastic transitions] For all $x \in S$ and $a \in A$, assume that $P(\cdot|x,a)$ is absolutely continuous w.r.t. $\mu$ and Radon-Nikodym derivative of $P$ w.r.t. $\mu$ is bounded uniformly with bound $C_\mu$:

$$C_\mu \triangleq \sup_{x \in S, a \in A} \left\| \frac{dP(\cdot|x,a)}{d\mu} \right\|_\infty < +\infty.$$  

We first show that the MCIP given by Eqs. (6)–(7) satisfies Assumption 5. According to Lemma 1, there exists $R_{\text{max}} = v_{\text{max}}$ such that $R_t \in [-R_{\text{max}}, R_{\text{max}}]$. As $\Pi_t(K_{t-1}, d_t) + c_t(K_t, K_{t-1})$ is deterministic and bounded, $P_{\text{R}}(\cdot|x,a)$ is a deterministic function and its support is included in the bounded set $[-R_{\text{max}}, R_{\text{max}}]$ which is independent of $(K_{t-1}, d_t)$.

Now we show that the MCIP satisfies Assumption 6. First, according to (Munos and Szepesvári 2008), Assumption 6 is equivalent to assuming that the transition kernel admits a uniformly bounded density when $\mu$ is the Lebesgue measure over $S$. Then, if the demands of the MICP are continuous, according to Assumption 2, $P(dy|x,a)$ satisfies $P(dy|x,a) \leq L_d \|D_{\text{max}}\|$ and is thus uniformly bounded. If the demands are discrete, the proof is trivial as $P(y|x,a) \leq 1$ for all $x,y \in S$. Hence, $P(dy|x,a)$ is uniformly bounded and the MCIP satisfies Assumption 6.

Acknowledgments

This research is supported by MOE Tier II grant MOE2015-T2-2-148 ‘Practical algorithms for large-scale dynamic programming’, and by the National Research Foundation, Prime Minister’s Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme.

References

Ahmed S, King AJ, Parija G (2003) A Multi-Stage Stochastic Integer Programming Approach for Capacity Expansion under Uncertainty. *Journal of Global Optimization* 26(1):3–24, ISSN 0925-5001, 1573-2916, URL http://dx.doi.org/10.1023/A:1023062915106.

Angelus A, Porteus E, Wood SC (2000) Optimal sizing and timing of modular capacity expansions. *Graduate School of Business Research Paper* (1479R2).

Angelus A, Porteus EL (2002) Simultaneous Capacity and Production Management of Short-Life-Cycle, Produce-to-Stock Goods Under Stochastic Demand. *Management Science* 48(3):399–413, ISSN 0025-1909, URL http://dx.doi.org/10.1287/mnsc.48.3.399.7726.
Zhao, Haskell, and Cardin: Approximate Dynamic Programming Algorithm for Multi-Stage Capacity Investment Problems

Antos A, Szepesvari C, Munos R (2007) Value-Iteration Based Fitted Policy Iteration: Learning with a Single Trajectory. 330–337 (IEEE), ISBN 978-1-4244-0706-4, URL http://dx.doi.org/10.1109/ADPRL.2007.368207.

Antos A, Szepesvári C, Munos R (2008) Fitted Q-iteration in continuous action-space MDPs. Advances in Neural Information Processing Systems, 9–16.

Bartlett PL, Harvey N, Liaw C, Mehrabian A (2017) Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks. arXiv:1703.02930 [cs].

Beale M, Hagan M, Howard D (2018) Neural Network Toolbox™ (The MathWorks, Inc.).

Bertsekas DP (2012) Dynamic Programming and Optimal Control, volume II (Athena Scientific), 4th edition edition.

Bertsimas D, Tsitsiklis JN (1997) Introduction to Linear Optimization, volume 6 (Athena Scientific Belmont, MA).

Cardin MA, Deng Y, Sun C (2017a) Real options and flexibility analysis in design and management of one-way mobility on-demand systems using decision rules. Transportation Research Part C: Emerging Technologies 84:265–287, ISSN 0968090X, URL http://dx.doi.org/10.1016/j.trc.2017.08.006.

Cardin MA, Xie Q, Ng TS, Wang S, Hu J (2017b) An approach for analyzing and managing flexibility in engineering systems design based on decision rules and multistage stochastic programming. IISE Transactions 49(1):1–12, ISSN 2472-5854, 2472-5862, URL http://dx.doi.org/10.1080/0740817X.2016.1189627.

Choromanska A, Henaff M, Mathieu M, Arous GB, LeCun Y (2014) The Loss Surfaces of Multilayer Networks. arXiv:1412.0233 [cs].

Dong JX, Lee CY, Song DP (2015) Joint service capacity planning and dynamic container routing in shipping network with uncertain demands. Transportation Research Part B: Methodological 78:404–421, ISSN 01912615, URL http://dx.doi.org/10.1016/j.trb.2015.05.005.

Eberly JC, Van Mieghem JA (1997) Multi-factor Dynamic Investment under Uncertainty. Journal of Economic Theory 75(2):345–387, ISSN 0022-0531, URL http://dx.doi.org/10.1006/jeth.1996.2281.
Geng N, Jiang Z, Chen F (2009) Stochastic programming based capacity planning for semiconductor wafer fab with uncertain demand and capacity. *European Journal of Operational Research* 198(3):899–908, ISSN 03772217, URL http://dx.doi.org/10.1016/j.ejor.2008.09.029.

Haskell WB, Jain R, Kalathil D (2016) Empirical Dynamic Programming. *Mathematics of Operations Research* 41(2):402–429, ISSN 0364-765X, URL http://dx.doi.org/10.1287/moor.2015.0733.

Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. *Neural Networks* 2(5):359–366, ISSN 0893-6080, URL http://dx.doi.org/10.1016/0893-6080(89)90020-8.

Hu YQ, Qian H, Yu Y (2017) Sequential Classification-Based Optimization for Direct Policy Search. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2029 (San Francisco, California USA: AAAI Press).

Huang K, Ahmed S (2009) The Value of Multistage Stochastic Programming in Capacity Planning Under Uncertainty. *Operations Research* 57(4):893–904, ISSN 0030-364X, URL http://dx.doi.org/10.1287/opre.1080.0623.

Jain AK, Mao J, Mohiuddin KM (1996) Artificial neural networks: A tutorial. *Computer* 29(3):31–44.

Jain R, Varaiya P (2010) Simulation-based optimization of Markov decision processes: An empirical process theory approach. *Automatica* 46(8):1297–1304, ISSN 00051098, URL http://dx.doi.org/10.1016/j.automatica.2010.05.021.

Laporte G, Louveaux FV (1993) The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters* 13(3):133–142, ISSN 0167-6377, URL http://dx.doi.org/10.1016/0167-6377(93)90002-X.

Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D (2015) Continuous control with deep reinforcement learning. *arXiv:1509.02971 [cs, stat]*.

Lu L, Yan X (2016) Capacity investment decisions under risk aversion: Capacity Investment Decisions. *Naval Research Logistics (NRL)* 63(3):218–235, ISSN 0894069X, URL http://dx.doi.org/10.1002/nav.21686.
Luss H (1982) Operations Research and Capacity Expansion Problems: A Survey. *Operations Research* 30(5):907–947, ISSN 0030-364X, URL [http://dx.doi.org/10.1287/opre.30.5.907](http://dx.doi.org/10.1287/opre.30.5.907).

Manne AS (1961) Capacity Expansion and Probabilistic Growth. *Econometrica* 29(4):632–649, ISSN 0012-9682, URL [http://dx.doi.org/10.2307/1911809](http://dx.doi.org/10.2307/1911809).

Martínez-Costa C, Mas-Machuca M, Benedito E, Corominas A (2014) A review of mathematical programming models for strategic capacity planning in manufacturing. *International Journal of Production Economics* 153:66–85, ISSN 0925-5273, URL [http://dx.doi.org/10.1016/j.ijpe.2014.03.011](http://dx.doi.org/10.1016/j.ijpe.2014.03.011).

Munos R, Szepesvári C (2008) Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research* 9(May):815–857.

Powell WB (2011) *Approximate Dynamic Programming: Solving the Curses of Dimensionality, 2nd Edition* (Hoboken, N.J: Wiley), 2 edition edition, ISBN 978-0-470-60445-8.

Schulman J, Moritz P, Levine S, Jordan M, Abbeel P (2015) High-Dimensional Continuous Control Using Generalized Advantage Estimation. *arXiv:1506.02438 [cs]*.

Sen S, Cochran JJ, Cox LA, Keskinocak P, Kharoufeh JP, Smith JC (2010) Stochastic Mixed-Integer Programming Algorithms: Beyond Benders’ Decomposition. *Wiley Encyclopedia of Operations Research and Management Science* (John Wiley & Sons, Inc.), ISBN 978-0-470-40053-1, URL [http://dx.doi.org/10.1002/9780470400531.eorms0232](http://dx.doi.org/10.1002/9780470400531.eorms0232).

Shapiro A, Dentcheva D, Ruszczyński AP (2009) *Lectures on Stochastic Programming: Modeling and Theory.* Number 9 in MPS-SIAM series on optimization (Philadelphia: Society for Industrial and Applied Mathematics : Mathematical Programming Society), ISBN 978-0-89871-687-0, oCLC: ocn402540716.

Singh KJ, Philpott AB, Wood RK (2009) Dantzig-Wolfe Decomposition for Solving Multistage Stochastic Capacity-Planning Problems. *Operations Research* 57(5):1271–1286, ISSN 0030-364X, 1526-5463, URL [http://dx.doi.org/10.1287/opre.1080.0678](http://dx.doi.org/10.1287/opre.1080.0678).
Sonoda S, Murata N (2015) Neural Network with Unbounded Activation Functions is Universal Approximator. arXiv:1505.03654 [cs, math].

Sun Y, Schonfeld P (2015) Stochastic capacity expansion models for airport facilities. Transportation Research Part B: Methodological 80:1–18, ISSN 01912615, URL http://dx.doi.org/10.1016/j.trb.2015.06.009.

Taghavi M, Huang K (2016) A multi-stage stochastic programming approach for network capacity expansion with multiple sources of capacity. Naval Research Logistics (NRL) 63(8):600–614, ISSN 1520-6750, URL http://dx.doi.org/10.1002/nav.21726.

Taghavi M, Huang K (2018) A Lagrangian relaxation approach for stochastic network capacity expansion with budget constraints. Annals of Operations Research ISSN 0254-5330, 1572-9338, URL http://dx.doi.org/10.1007/s10479-018-2862-7.

Trigeorgis L (1996) Real Options: Managerial Flexibility and Strategy in Resource Allocation (MIT Press), ISBN 978-0-262-20102-5.

Truong VA, Roundy RO (2011) Multidimensional Approximation Algorithms for Capacity-Expansion Problems. Operations Research 59(2):313–327, ISSN 0030-364X, 1526-5463, URL http://dx.doi.org/10.1287/opre.1100.0892.

Van Mieghem JA (2003) Commissioned Paper: Capacity Management, Investment, and Hedging: Review and Recent Developments. Manufacturing & Service Operations Management 5(4):269–302, ISSN 1523-4614, URL http://dx.doi.org/10.1287/msom.5.4.269.24882.

Ye Q, Duenyas I (2007) Optimal Capacity Investment Decisions with Two-Sided Fixed-Capacity Adjustment Costs. Operations Research 55(2):272–283, ISSN 0030-364X, 1526-5463, URL http://dx.doi.org/10.1287/opre.1060.0386.

Zhao S, Haskell WB, Cardin MA (2017) An Approximate Dynamic Programming Approach for Multi-Facility Capacity Expansion Problem with Flexibility Design. IIE Annual Conference. Proceedings; Norcross 440–445.

Zhao S, Haskell WB, Cardin MA (2018) Decision rule-based method for flexible multi-facility capacity expansion problem. IIEE Transactions 50(7):553–569, ISSN 2472-5854, URL http://dx.doi.org/10.1080/24725854.2018.1426135.
Zou J, Ahmed S, Sun XA (2018) Stochastic dual dynamic integer programming. *Mathematical Programming* ISSN 0025-5610, 1436-4646, URL http://dx.doi.org/10.1007/s10107-018-1249-5.