Deep Value Function Networks for Large-scale Multistage Stochastic Programs

Hyunglip Bae
qogudflq@kaist.ac.kr
KAIST

Jinkyu Lee
jinkyu.lee@hugraph.com
Hugraph

Woo Chang Kim[*]
wkim@kaist.ac.kr
KAIST

Yongjae Lee[*]
yongjaelee@unist.ac.kr
UNIST

Abstract

A neural networks-based stagewise decomposition algorithm called Deep Value Function Networks (DVFN) is proposed for large-scale multistage stochastic programming (MSP) problems. Traditional approaches such as nested Benders decomposition and its stochastic variant, stochastic dual dynamic programming (SDDP) approximates value functions as piecewise linear convex functions by gradually accumulating subgradient cuts from dual solutions of stagewise subproblems. Although they have been proven effective for linear problems, nonlinear problems may suffer from the increasing number of subgradient cuts as they proceed. A recently developed algorithm called Value Function Gradient Learning (VFGL) replaced the piecewise linear approximation with parametric function approximation, but its performance heavily depends upon the choice of parametric forms like most of traditional parametric machine learning algorithms did. On the other hand, DVFN approximates value functions using neural networks, which are known to have huge capacity in terms of their functional representations. The art of choosing appropriate parametric form becomes a simple labor of hyperparameter search for neural networks. However, neural networks are non-convex in general, and it would make the learning process unstable. We resolve this issue by using input convex neural networks that guarantee convexity with respect to inputs. We compare DVFN with SDDP and VFGL for solving large-scale linear and nonlinear MSP problems: production optimization and energy planning. Numerical examples clearly indicate that DVFN provide accurate and computationally efficient solutions.

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206. Copyright 2023 by the author(s).

1 INTRODUCTION

Sequential decision-making problems under uncertainty are addressed with various theories including stochastic control, reinforcement leaning (RL), and multistage stochastic programming (MSP). Stochastic control usually tries to find and analyze analytic solutions based on carefully defined random variables and their dynamics in a continuous-time setting. On the other hand, both RL and MSP approximate the problem and underlying uncertainties within a discrete-time setting. RL learns in a trial-and-error manner mostly in discretized state and action spaces, while MSP is based on mathematical formulations that represent every consequence of actions that are usually continuous. Therefore, RL would be more suitable when the environment can be simulated, and MSP would be suitable when the environment can be well represented by mathematical formulas.

In this paper, we focus on the latter case (i.e., MSP), where production optimization, hydropower production planning, and asset liability management (Shiina and Birge, 2003; Carino et al., 1994; Fleten and Kristoffersen, 2008) are typical examples. MSP incorporates the uncertain environment via scenario trees that approximates the underlying stochastic process by a set of finite scenarios with discretized time points. Then, the original stochastic problem can be transformed into a single large deterministic equivalent problem. However, deterministic equivalent problems often become computationally intractable because the number of scenarios increases exponentially as the number of stages and/or the number of nodes per stage increase. So called "the curse of dimensionality" is a critical drawback of MSP when dealing with large-scale problems.

Various decomposition-based algorithms are proposed to resolve the curse of dimensionality of large-scale MSP problems. There are mainly two types of decomposition algorithms: stagewise and scenario decomposition. They have different approaches to handle nonanticipativity constraints, which make sure that decisions are made without anticipating what is going to happen in the future.

For the stagewise decomposition approach, stochastic programs are broken down into stagewise subproblems and they are sequentially solved. For each subproblem, the effect of immediate decision on later stages is reflected by the "value function". Even for a simple problem, it is extremely difficult to find the exact explicit form of the value function. Thus, the main goal of stage decomposition methods is to approximate the value function as accurately and efficiently as possible. The most popular stagewise decomposition algorithm would be the stochastic dual dynamic programming (SDDP) (Pereira and Pinto, 1991), which is an extension of nested Benders decomposition. Here, value functions are approximated as piecewise linear convex functions, which are constructed gradually based on the dual solution of each subproblem. While the SDDP is guaranteed to converge to the optimal solution under mild conditions, its computational time increases every iteration because the size of piecewise linear approximation increases monotonically.

For the scenario decomposition approach, stochastic programs are decomposed scenario-wise, and thus, the nonanticipativity constraint can be relaxed within each subproblem. Such decomposition algorithms include the dual decomposition algorithm (Carøe and Schultz, 1999) and progressive hedging (Rockafellar and Wets, 1991) which have been widely adopted for various large-scale stochastic programs (Triki et al., 2005; Berg et al., 2014; Fadda et al., 2019). However, in order to obtain solutions that satisfy the nonanticipativity constraints after combining subproblems, every scenario should be considered at every iteration. Therefore, the scenario decomposition methods would be inefficient for extremely large-scale MSP problems.

A recently proposed stagewise decomposition algorithm called "value function gradient learning" (VFGL) (Lee et al., 2022) approximates the value function within a predetermined parametric function form. It learns parameters by using the gradient information of the true value function as in SDDP. Since the value function is approximated with a fixed parametric form, the size of subproblems remains almost constant at every iteration. Thus, VFGL is computationally stable, even for large iterations. However, the performance of VFGL is heavily dependent upon the choice of parametric form, and it affects the convergence to the optimality. This is a common problem for most parametric machine learning models, and the choice of good parametric form is often regarded as an "art".

In this paper, we propose a neural networks based stagewise decomposition algorithm called "deep value function network" (DVFN), which replaces the parametric approximation of VFGL using neural networks. Given neural networks' extreme capability in function approximation, users do not need an artistic sense of choosing the appropriate functional form. Instead, DVFN requires an ordinary hyperparameter search for neural networks. Just like parametric machine learning models have evolved into deep learning models, algorithms for stochastic optimization are evolving in the same direction. One concern would be that neural networks are non-convex with respect to inputs in general, and it would severely disturb the stability of the stagewise decomposition. Hence, DVFN uses input convex neural networks (Amos et al., 2017) which are scalar-valued neural networks that are always convex functions with respect to the inputs. The approximation of value function of SDDP, VFGL, and DVFN are illustrated in Figure 1.

Here we note the difference between DVFN and the Neural SDDP (ν-SDDP) proposed by Dai et al. (2021) and Neural Two-stage Stochastic Programming (Neur2SP) proposed by Dumouchelle et al. (2022). While ν-SDDP and Neur2SP also use neural networks to extend stochastic programming, but its purpose is clearly different from DVFN (and SDDP and VFGL as well). They both learns how to solve a specific MSP problem from a large number of actual runs of SDDP or 2SP. They are useful when MSP problems with a fixed form are needed to be solved over and over again with slightly changing settings. This can happen in practice when a problem should be solved on a daily basis or when a problem needs to be solved for each of a large number of customers. Of course, if one wants to solve a different type of problem (e.g., different set of constraints, different number of decision variables), ν-SDDP and Neur2SP need to be trained again with a large number of SDDP or 2SP runs. Therefore, in terms of solving an arbitrary stochastic programming problem, ν-SDDP and Neur2SP would be extremely inefficient, and thus, it would not be appropriate to directly compare ν-SDDP or Neur2SP with SDDP, VFGL, and DVFN.

The remainder of the paper is organized as follows. First, we provide the background of SDDP and VFGL with problem formulations in Section 2. DVFN is derived in Section 3. Input convex neural networks, quasi-Newton based primal-dual algorithm, and stopping criterion are described. Section 4 provides numerical experiments that compare SDDP, VFGL and DVFN approaches for solving large-scale production optimization and hydrothermal scheduling problems. Finally, Section 5 concludes the paper.

2 PRELIMINARY

We begin by formally defining the multistage stochastic programming problem. Then, we will briefly introduce SDDP and VFGL before presenting DVFN in the next section.

2.1 Problem Setting

We consider a sequential decision-making problem with multiple time periods where the stochastic process that governs underlying uncertainties is gradually observed. Denote the stochastic process as $\xi_{[T]} = (\xi_1, \ldots, \xi_T)$ and the decision process $x_{[T]} = (x_1, \ldots, x_T)$ where ξ_1 is deterministic and $x_t \in R^{n_t}$. Here, it is assumed that each ξ_t has finite moments and that each x_t is determined with only the information up to time t. Specifically, the two processes proceed

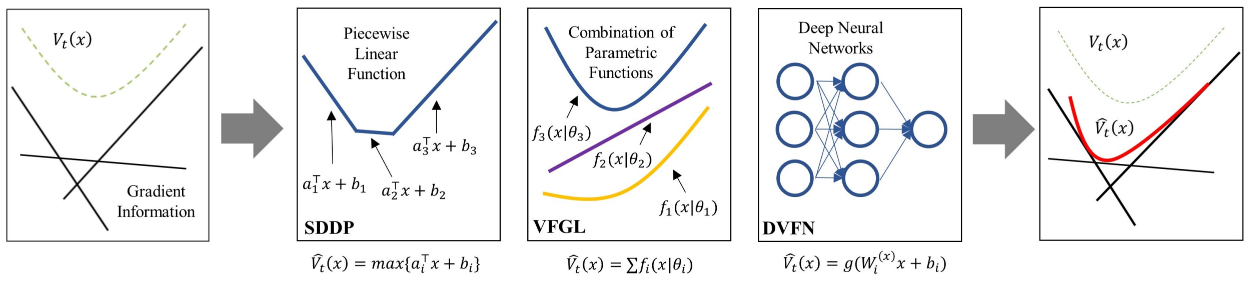


Figure 1: Construction of approximation in each algorithm

alternately according to the following sequence (Shapiro et al., 2021):

$$x_1 \rightsquigarrow \xi_2 \rightsquigarrow x_2 \rightsquigarrow \ldots \rightsquigarrow \xi_T \rightsquigarrow x_T$$

A nested formulation of T-stage stochastic program is as follows.

$$\min_{x_1 \in \mathcal{X}_1} f_1(x_1) + \mathbb{E}[\min_{x_2 \in \mathcal{X}_2(x_1,\xi_2)} f_2(x_2,\xi_2) \tag{1}$$
$$+ \mathbb{E}_{\cdot|\xi_{[2]}}[\ldots + \mathbb{E}_{\cdot|\xi_{[T-1]}}[\min_{x_T \in \mathcal{X}_T(x_{T-1},\xi_T)} f_T(x_T,\xi_T)]]]$$

where $\mathbb{E}_{\cdot|\xi_{[t]}}$ is the conditional expectation with respect to $\xi_{[t]}$, $f_t(x_t,\xi_t)$ is a convex objective function in x_t, and $\mathcal{X}_t(x_{t-1},\xi_t)$ is a convex feasible region for x_t given x_{t-1} and ξ_t. Specifically, the feasible region is an intersection of sublevel sets of convex functions and hyperplanes, which is expressed as $\mathcal{X}_t(x_{t-1},\xi_t) = \{x_t : g_{t,i}(x_t,\xi_t) \leq -h_{t,i}(x_{t-1},\xi_t), i = 1,\ldots,p_t\} \cap \{x_t : l_{t,j}(x_t,\xi_t) = b_{t,j}(x_{t-1},\xi_t), j = 1,\ldots,q_t\}$, where $g_{t,i}$ is a twice-differentiable convex function in x_t, $l_{t,j}$ is a linear function in x_t, $h_{t,j}$ is a twice-differentiable convex function in x_{t-1} and $b_{t,j}$ is a linear function in x_{t-1}.

The usual MSP approach solves (1) by constructing a large deterministic equivalent convex optimization problem with a scenario tree which approximate the stochastic process $\xi_{[T]}$ with finite realization. However, the constructed problem often becomes intractable, and thus, stagewise decomposition algorithms decompose the problem into the following stagewise subproblems.

For $t = 1$,

$$\min_{x_1 \in \mathcal{X}_1} \quad f_1(x_1) + V_2(x_1)$$

For $t = 2,\ldots,T$,

$$\min_{x_t \in \mathcal{X}_t(x_{t-1},\xi_t)} \quad f_t(x_t,\xi_t) + V_{t+1}(x_t,\xi_{[t]}) \tag{2}$$

where the value function V_t is recursively defined by the following Bellman Equation.

For $t = T,\ldots,2$,

$$\mathcal{V}_t(x_{t-1},\xi_{[t]}) = \inf_{x_t \in \mathcal{X}_t(x_{t-1},\xi_t)} \{f_t(x_t,\xi_t) + V_{t+1}(x_t,\xi_{[t]})\}$$
$$V_{t+1}(x_t,\xi_{[t]}) = \mathbb{E}_{\cdot|\xi_{[t]}}[\mathcal{V}_{t+1}(x_t,\xi_{[t+1]})] \tag{3}$$

with $V_{T+1} = 0$.

Most stagewise decomposition algorithms make the following assumptions.

1) Stagewise independence: For stage $t = 1,\ldots,T-1$, ξ_{t+1} is independent of $\xi_{[t]}$

2) Relatively complete recourse: For stage $t = 2,\ldots,T$ for any previous stage solution x_{t-1}, and for any observation ξ_t, the feasible region $\mathcal{X}_t(x_{t-1},\xi_t)$ is bounded and nonempty.

If ξ_{t+1} is independent of $\xi_{[t]}$, the conditional expectation operator in (3) becomes redundant, i.e. $V_{t+1}(x_t,\xi_{[t]})$ does not depend on $\xi_{[t]}$. Therefore, $V_{t+1}(x_t,\xi_{[t]})$ is equivalent to $V_{t+1}(x_t)$

2.2 Stochastic Dual Dynamic Programming

Stochastic Dual Dynamic Programming (SDDP) (Pereira and Pinto, 1991) is one of the most popular stagewise decomposition methods for large-scale multistage stochastic programming problems. SDDP solves (2) by approximating V_t as a convex piecewise linear function based on Benders decomposition. Each iteration of SDDP consists of forward and backward pass.

In forward pass starting from $t = 0$, trial solutions of each stage are obtained by solving stagewise subproblems using current approximation of future value function V_{t+1}. Specifically, for the n-th iteration of stage t with sample ξ_t^s, a trial solution x_t can be obtained as follows.

$$x_t = \operatorname*{arg\,min}_{x_t \in \mathcal{X}_t(x_{t-1},\xi_t^s)} f_t(x_t,\xi_t^s) + V_{t+1}^n(x_t)$$

Here, V_{t+1}^n is the current approximation of value function V_{t+1} which is expressed as

$$V_{t+1}^n(x_t) = \max_{k=1,\ldots,n} \{(\beta_{t+1}^k)^\top x_t + \alpha_{t+1}^k\}$$

where β_{t+1}^k and α_{t+1}^k are calculated from backward pass.

Then, we can transform the stage t subproblem into an equivalent form without value function V_{t+1} as

$$\begin{aligned} \min_{x_t \in \mathcal{X}_t(x_{t-1}, \xi_t^s)} \quad & f_t(x_t, \xi_t^s) + \theta_{t+1} \\ \text{s.t} \quad & -(\beta_{t+1}^k)^\top x_t + \theta_{t+1} \geq \alpha_{t+1}^k, k = 1, \ldots, n \end{aligned}$$

In backward pass starting from $t = T$, the approximated value function V_t^n is updated by adding linear cut (linear constraints) derived from optimal solutions of stage t subproblem. Specifically, for the n-th iteration of stage t with sample ξ_t^s, the approximated value function V_t^n is updated into V_t^{n+1} using primal and dual optimal solutions (x_t^*, u_t^*, v_t^*) of the stage t subproblem as follows.

$$V_t^{n+1}(x_{t-1}) = \max\{V_t^n(x_{t-1}), (\beta_t^{n+1})^\top x_{t-1} + \alpha_t^{n+1}\}$$

where

$$\begin{aligned} \beta_t^{n+1} = \frac{1}{S} \sum_{s=1}^{S} (& \sum_{i=1}^{p_t} u_{t,i}^* \nabla_{x_{t-1}} h_{t,i}(x_{t-1}, \xi_t^s) \\ & + \sum_{j=1}^{q_t} v_{t,j}^* \nabla_{x_{t-1}} b_{t,j}(x_{t-1}, \xi_t^s)) \end{aligned}$$

$$\alpha_t^{n+1} = \frac{1}{S} \sum_{s=1}^{S} f_t(x_t^*, \xi_t^s) + V_{t+1}^n(x_t^*)$$

2.3 Value Function Gradient Learning

Value Function Gradient Learning (VFGL) (Lee et al., 2022) is a gradient-learning based algorithm for large-scale multistage stochastic programming problems. The main idea of VFGL is to approximate the value function $V_t(x_{t-1})$ as a specific parametric convex function $\hat{V}_t(x_{t-1}, \theta_t)$ instead of piecewise linear functions as in SDDP. It finds parameter θ_t by matching the gradient of the parametric function to that of the true value function by utilizing the duality of optimization problems (Boyd et al., 2004). More specifically, it can utilize a stochastic gradient descent type of algorithm to minimize the difference between the gradients of the parametric function and the true value function.

As in many parametric machine learning models, it is crucial to choose an appropriate parametric function form for the model to perform well. Lee et al. (2022) suggested that the indefinite integral form of the stagewise objective function would be a good candidate to start with. They showed that a sufficiently good optimal solution can be obtained if an appropriate parametric function form is used.

3 DEEP VALUE FUNCTION NETWORKS

Although VFGL has many computational advantages for solving large-scale multistage stochastic optimization problems, it requires an appropriate choice of convex parametric family to approximate the value function. Even a person familiar with the structure of the optimization problem, finding an appropriate parametric form is extremely difficult and new parameter form must be set for other types of problems. As in many other parametric machine learning algorithms, choosing the right parametric form is very tricky. To overcome this limitation, DVFN replaces parametric function approximations of VFGL by approximations via neural networks.

Both DVFN and VFGL approximate value functions by using their gradient information. More specifically, the gradient learning approach is motivated from the Karush-Kuhn-Tucker (KKT) conditions (Boyd et al., 2004). The KKT conditions are necessary conditions for optimization problems with differentiable objective and constraint functions that achieve strong duality, but they become sufficient conditions for convex optimization problems. Consider a stage t subproblem of (2) where $V_{t+1}(x_t)$ is replaced by approximation $\hat{V}_{t+1}(x_t, \theta_{t+1})$.

$$\min_{x_t \in \mathcal{X}_t(x_{t-1}, \xi_t)} \quad f_t(x_t, \xi_t) + \hat{V}_{t+1}(x_t, \theta_{t+1}) \tag{4}$$

Then its KKT conditions are as follows.

1) Stationarity:

$$\begin{aligned} & \nabla_{x_t} f_t(x_t, \xi_t) + \nabla_{x_t} \hat{V}_{t+1}(x_t, \theta_{t+1}) \\ & + \sum_{i=1}^{p_t} u_{t,i} \nabla_{x_t} (g_{t,i}(x_t, \xi_t) + h_{t,i}(x_{t-1}, \xi_t)) \\ & + \sum_{j=1}^{q_t} v_{t,j} \nabla_{x_t} (l_{t,j}(x_t, \xi_t) - b_{t,j}(x_{t-1}, \xi_t)) = 0 \end{aligned}$$

2) Primal and dual feasibility: For $i = 1, \ldots, p_t$, $j = 1, \ldots, q_t$,

$$\begin{aligned} & g_{t,i}(x_t, \xi_t) \leq -h_{t,i}(x_{t-1}, \xi_t) \\ & l_{t,j}(x_t, \xi_t) = b_{t,j}(x_{t-1}, \xi_t), \quad u_{t,i} \geq 0 \end{aligned}$$

3) Complementary slackness: For $i = 1, \ldots, p_t$,

$$u_{t,i} \nabla_{x_t} (g_{t,i}(x_t, \xi_t) + h_{t,i}(x_{t-1}, \xi_t)) = 0$$

Let $(\hat{x}_t^*, \hat{u}_t^*, \hat{v}_t^*)$ be a solution to the KKT conditions. Then it is the optimal solution to the stage t subproblem, in which V_t is replaced by $\hat{V}_t$. Since the approximated problem has the same set of constraints, $(\hat{x}_t^*, \hat{u}_t^*, \hat{v}_t^*)$ still satisfies the primal feasibility, dual feasibility and complementary slackness of original problem, while left side of the stationarity condition become

$$\begin{aligned}&\nabla_{x_t} f_t(\hat{x}_t^*, \xi_t) + \nabla_{x_t} V_{t+1}(\hat{x}_t^*)\\ &+ \sum_{i=1}^{p_t} \hat{u}_{t,i}^* \nabla_{\hat{x}_t^*}(g_{t,i}(\hat{x}_t^*, \xi_t) + h_{t,i}(x_{t-1}, \xi_t))\\ &+ \sum_{j=1}^{q_t} \hat{v}_{t,i}^* \nabla_{\hat{x}_t^*}(l_{t,j}(\hat{x}_t^*, \xi_t) - b_{t,j}(x_{t-1}, \xi_t))\\ &= \nabla_{x_t} V_{t+1}(\hat{x}_t^*) - \nabla_{x_t} \hat{V}_{t+1}(\hat{x}_t^*, \theta_{t+1})\end{aligned}$$

Given this insight, DVFN approximates the value function with neural networks by minimizing the squared sum of gradient error of each stagewise subproblem expressed as $\|\nabla_{x_t} V_{t+1}(\hat{x}_t^*) - \nabla_{x_t} \hat{V}_{t+1}(\hat{x}_t^*, \theta_{t+1})\|^2$. Note that Lee et al. (2022) showed for VFGL that the approximated solution $(\hat{x}_t^*, \hat{u}_t^*, \hat{v}_t^*)$ converges to the true optimal solution as this error term approaches zero, and this can be directly applied to DVFN with input convex neural networks as well.

3.1 Input Convex Neural Networks

It is well known that any continuous function can be closely approximated by neural networks from the universal approximate theorem (Cybenko, 1989; Hornik et al., 1989; Funahashi, 1989). However, for stability of the algorithm, it is necessary to keep the neural networks convex to the input, since the value function V_{t+1} in Problem (2) is convex under mild conditions (see Appendix A for proof). We stabilize DVFN algorithm by using input convex neural networks (ICNN) that have special network architecture to ensure convexity with respect to inputs. To ensure convexity, certain constraints on weights and activation functions are required. In Appendix E, we compare the performance of DVFN with plain feedforward networks and DVFN with ICNN. It clearly indicates that maintaining the convexity of neural networks is actually helpful for the algorithm convergence.

Amos et al. (2017) proposed two kind of ICNNs, fully input convex neural networks (FICNN) and partially input convex neural networks (PICNN). Since we need the convexity with respect to the entire input space, FICNN is used. For $i = 0, \ldots, l-1$, FICNN is described by

$$y_{i+1} = g_i(W_i^{(y)} y_i + W_i^{(x)} x + b_i), \quad F(x, \theta) = y_l$$

where x is the input features, y_{i+1} denotes the output of the i-th layer (with $y_0, W_0^{(y)} = 0$), $\theta = \{W_{0:l-1}^{(y)}, W_{1:l-1}^{(x)}, b_{0:l-1}\}$ are weights of networks, g_i are activation functions, and F denotes FICNN. All weights $W_{0:l-1}^{(y)}$ are constrained to be non-negative, and all activation functions g_i's are restricted to be non-decreasing and convex. Then, the output y_i is convex with respect to the input features x. The convexity of FICNN follows from basic properties that non-negative sums of convex functions and compositions of a convex function and a convex non-decreasing function are also convex. Notable difference compared to vanilla feedforward neural networks is that 'pass-through (or skip)' connections from the input layer to hidden layers are introduced. That is, y_i depends on the input x as well as the output of the previous layer y_{i-1}.

3.2 Primal-dual interior point method with ICNN

Consider the stage t subproblem of Problem (4) where $\hat{V}_{t+1}(x_t, \theta_{t+1})$ is represented by FICNN. This subproblem can be solved by the primal-dual interior point method (Wright, 1997). The perturbed KKT conditions are given as follows.

$$\begin{aligned}r_{dual} &= \nabla f_t(x_t, \xi_t) + \nabla \hat{V}_{t+1}(x_t, \theta_{t+1})\\ &\quad + \nabla g_t(x_t, \xi_t) u_t + \nabla l_t(x_t, \xi_t) v_t = 0,\\ r_{cent} &= U_t(g_t(x_t, \xi_t) + h_t(x_{t-1}, \xi_t)) + \tau \mathbf{1} = 0,\\ r_{prim} &= l_t(x_t, \xi_t) - b_t(x_{t-1}, \xi_t) = 0,\\ g_t(x_t, \xi_t) &\le -h_t(x_{t-1}, \xi_t), \quad u_t > 0\end{aligned}$$

where $\nabla g_t(x_t, \xi_t) = [\nabla g_{t,1}(x_t, \xi_t), \ldots, \nabla g_{t,p_t}(x_t, \xi_t)]$, $\nabla l_t(x_t, \xi_t) = [\nabla l_{t,1}(x_t, \xi_t), \ldots, \nabla l_{t,q_t}(x_t, \xi_t)]$ and $U_t = diag(u_t)$. The perturbed KKT conditions , which are nonlinear equations, can be solved by approximating them into linear equations using the Newton's method. Define $GH_t = diag(g_t(x_t, \xi_t) + h_t(x_{t-1}, \xi_t))$ and $r(x_t, u_t, v_t) = [r_{dual}, r_{cent}, r_{prim}]^\top$.

Then, the first order Taylor approximation of $r(x_t + \Delta x_t, u_t + \Delta u_t, v_t + \Delta v_t)$ is

$$\begin{aligned}&r(x_t + \Delta x_t, u_t + \Delta u_t, v_t + \Delta v_t)\\ &\approx r(x_t, u_t, v_t) + \nabla r(x_t, u_t, v_t) \begin{bmatrix}\Delta x_t\\ \Delta u_t\\ \Delta v_t\end{bmatrix},\end{aligned}$$

and we can find $(\Delta x_t, \Delta u_t, \Delta v_t)$ by solving

$$\nabla r(x_t, u_t, v_t) \begin{bmatrix}\Delta x_t\\ \Delta u_t\\ \Delta v_t\end{bmatrix} = -r(x_t, u_t, v_t).$$

The primal-dual interior point method finds the optimal solution by gradually updating the primal and dual solution using Δx_t, Δu_t, and Δv_t. The step size γ for the update can be determined by the backtracking line search. However, calculating the Hessian $\nabla^2 \hat{V}_{t+1}$ of FICNN is computationally expensive. Thus, we approximate the Hessian with a matrix B_t and updated it using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update (Fletcher, 1987). In BFGS update, B_t is initialized with an identity matrix and updated by the rule

$$B_t \leftarrow B_t - \frac{B_t s_t s_t^\top B_t}{s_t^\top B_t s_t} + \frac{y_t y_t^\top}{y_t^\top s_t}$$

where $y_t = \nabla \hat{V}_{t+1}(x_t^+) - \nabla \hat{V}_{t+1}(x_t)$, and $s_t = x_t^+ - x_t = \gamma \Delta x_t$.

3.3 Sobolev training and loss function

When approximating a function with neural networks, the loss function is usually designed to make the neural networks to output a value that is close to the value of the target function for a given input. However, VFGL and DVFN approximates value functions based on their gradient information, not outputs, so the loss function should incorporate the difference between the gradient of FICNN and that of the target function. Czarnecki et al. (2017) proposed the Sobolev training, which approximates not only the output of the function but also its gradient. Specifically, suppose that when learning a function F from input x_i, we know not only the function value $F(x_i)$ but also the j-th order derivative $\nabla^j F(x_i)$. In other words, typical training data pairs $\{(x_i, F(x_i))\}_{i=1}^N$ can be extended to $\{(x_i, F(x_i), \nabla F(x_i), \nabla^2 F(x_i), \ldots, \nabla^K F(x_i))\}_{i=1}^N$. For our problem, the Sobolev training objective would be to minimize

$$\sum_{i=1}^{N}[L(\hat{F}(x_i,\theta), F(x_i)) + \sum_{j=1}^{K} L_j(\nabla^j \hat{F}(x_i,\theta), \nabla^j F(x_i))]$$

where $\hat{F}(x,\theta)$ is FICNN parameterized by θ, and L_j is a loss function for the j-th order derivative. Czarnecki et al. (2017) showed that minimizing this loss function is effective for complex learning. Since DVFN considers only the first order derivative information, we use Sobolev training with $K = 1$ without L, while L_1 is set as the typical mean-square error loss function.

3.4 Stopping criterion

The convergence of weights of FICNN can be used as the stopping criterion of DVFN. In particular, the total weight change in the i-th iteration can be defined as $\Delta\theta^i = \sum_{t=2}^{T} \Delta\theta_t^i$, where $\Delta\theta_t^i = \|\theta_t^i - \theta_t^{i-1}\|$ is the change of weights of $\hat{V}_{t+1}$ in the i-th iteration. One can conclude that the DVFN converges when $\Delta\theta^i$ becomes sufficiently small.

The entire procedure of DVFN is presented in Algorithm 1.

Algorithm 1 Deep Value Function Networks

```
Require: $N$ (maximum iteration), $S$ (sample size), $\varepsilon$ (stopping threshold), $\hat{V}_{t+1}$ for $t = 1, \ldots, T-1$ (ICNNs)
Ensure: $i \leftarrow 1$ (iteration counter), $\Delta\theta^i \leftarrow \infty$ (total theta update level)
  while $\Delta\theta^i > \varepsilon$ and $i < N$ do
    $\Delta\theta^i \leftarrow 0$
    for $t = 1, \ldots, T$ do
      if $t = 1$ then
        Solve stage 1 subproblem 2
        and save optimal solution $x_1^i$
      else if $t > 1$ then
        Sample random variables $\xi_t^s$ for $s = 1, \ldots, S$
        for $s = 1, \ldots, S$ do
          Solve stage $t$ subproblem 2
          and sample $\nabla_{x_{t-1}} \mathcal{V}_t(x_{t-1}, \xi_t^s)$
        end for
        Train $\hat{V}_t$ with sample $(x_{t-1}^i, \nabla_{x_{t-1}} V_t(x_{t-1}))$
        and save $x_t^i$ for random sample $\xi_t^{s'}$
        $\Delta\theta^i \leftarrow \Delta\theta^i + \|\theta_t^{i+1} - \theta_t^i\|$
      end if
    end for
    $i \leftarrow i + 1$
  end while
```

4 NUMERICAL EXPERIMENTS

In this section, we present numerical experiments. Two popular problems in optimization community are considered: production optimization and energy planning. Note that these two problems can be easily extended to many other operations research problems. Both are formulated as convex multistage stochastic programs. Each problem is solved using four different approaches: MSP, SDDP, VFGL, and DVFN. Here, we used a large-scale scenario tree for the MSP approach without considering computational cost so that it can provides a benchmark solution. Hence, we can see the optimality of the solutions from SDDP, VFGL, and DVFN by comparing them with the solution from MSP.

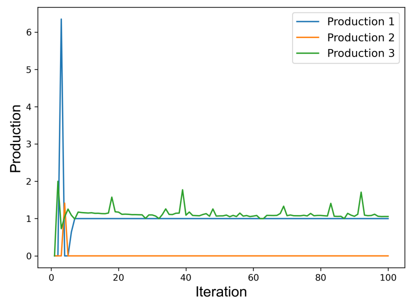
For VFGL, we use three different types of parametric forms (exponential, quadratic, and linear) so that we can see the performance change depending on the choice of parametric form. The hyperparameters for FICNN in DVFN are tuned as in Schalbetter (2020). More details are shown in Appendix D. All the experiments were done using an Intel i5-9400F processor with 64 GB of RAM and GeForce RTX 3070. [1].

We compare algorithms based on three aspects: objective function value, first stage solutions, and computation time. We present averaged values and their standard errors after 20 repeated experiments.The purpose of the algorithms is to obtain an objective function value that is as close as possible to that of MSP (oracle) with low computational cost. In MSP studies, first stage solutions are often analyzed in detail, because they are the solutions that would be actually deployed when MSP is used for real-world decision-making problems.

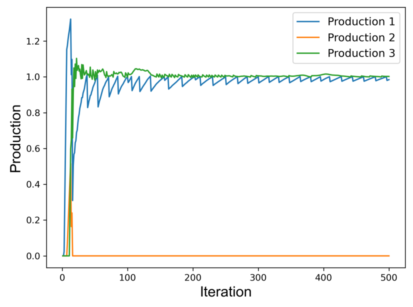
4.1 Production optimization

We consider an 11-stage factory production optimization problem with 3 scenario branches, in which the objective is to determine the optimal production amount of three products at each stage to satisfy random demands while minimizing the sum of manufacturing, delivery, and inventory

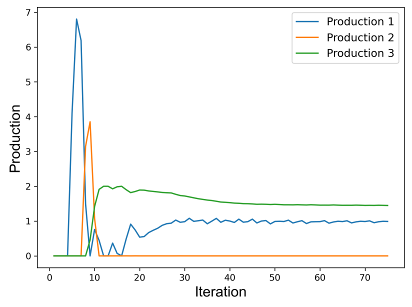
[1]Codes for DVFN and all of our experiments can be found at: `https://github.com/HyunglipBae/DVFN`



(a) SDDP

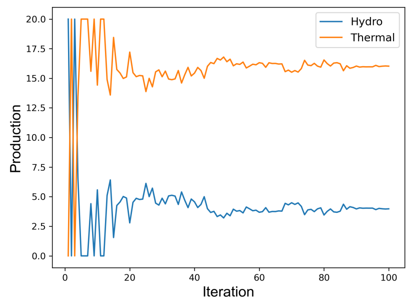


(b) VFGLexp

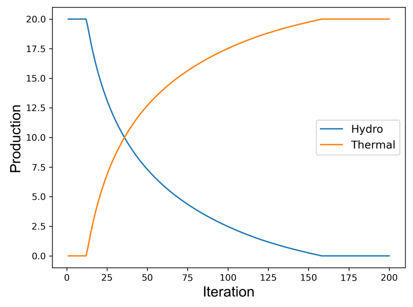


(c) DVFN

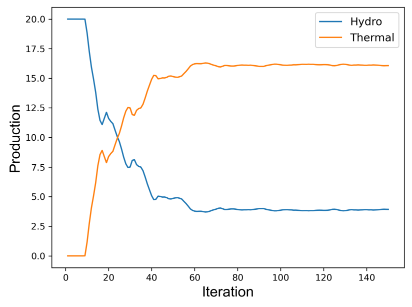
Figure 2: First stage decisions of different algorithms for production optimization for each iteration



(a) SDDP



(b) VFGLexp



(c) DVFN

Figure 3: First stage decisions of different algorithms for energy planning for each iteration

Table 1: Performance comparisons of algorithms for production optimization

Algorithm	Objective	Production 1	Production 2	Production 3	Time (s)
MSP	210	1.00	0.00	1.50	10215
SDDP	210 (0.30)	1.00 (0.00)	0.00 (0.00)	1.05 (0.01)	1074 (4.67)
VFGLexp	210 (0.00)	0.99 (0.00)	0.00 (0.00)	1.03 (0.01)	798 (3.73)
VFGLquad	217 (0.00)	1.99 (0.00)	0.00 (0.00)	0.00 (0.00)	774 (1.13)
VFGLlinear	219 (0.00)	0.50 (0.49)	0.00 (0.00)	0.00 (0.00)	551 (2.66)
DVFN	210 (0.14)	0.99 (0.00)	0.00 (0.00)	1.45 (0.01)	574 (1.36)

costs. Variants of this problem has been extensively studied in various literature (e.g.(Wagner and Whitin, 1958; Shapiro, 1993; Karimi et al., 2003)). The formulation and details of decision variables and parameters are in Appendix B.1.

In VFGL, the following parametric form was used to reflect the insight that the optimal solution would create nonempty storage to avoid outsourcing and that the disutility of storage would increase at a certain point.

$$\hat{V}_t(s_{t-1}, \theta_t = (\theta_t^1, \theta_t^2)) = s_{t-1}\theta_t^1 + \sum_{i=1}^{3} e^{-\theta_{t,i}^2 s_{t-1,i}}$$

where $\theta_t^1, \theta_t^2 \in \mathbf{R}^3$. In this paper, the solutions from VFGL with following other parametric form are also compared.

$$\hat{V}_t^{quad}(s_{t-1}, \theta_t = (\theta_t^1, \theta_t^2)) = s_{t-1}\theta_t^1 + s_{t-1}^2\theta_t^2$$
$$\hat{V}_t^{linear}(s_{t-1}, \theta_t = (\theta_t^1, \theta_t^2)) = s_{t-1}\theta_t^1$$

The results are presented in Figure 2 and Table 3. First, Table 3 summarizes the average performances of SDDP, VFGL, and DVFN after 20 repeated experiments, where values in parentheses are standard errors. We can see that there is no substantial difference in objective values between different algorithms except for VFGL with linear and quadratic parametric form. It shows that VFGL may severly underperform when a bad parametric form is chosen. DVFN and VFGLexp achieve almost the same objective values with the benchmark (MSP). However, we can see that the standard error of DVFN is slightly larger than VFGLexp but smaller than SDDP. As for first stage solutions, DVFN is closest to MSP. SDDP and VFGLexp generate quite different from MSP, but considering the objective value, the seem to be another optimal solution. Also, Figure 2 shows that DVFN finds optimal solutions in a quite stable way compared to SDDP and VFGL. There are more fluctuations in solutions in SDDP and VFGL. Also, for computation time, we can see that DVFN is much better than MSP, SDDP. In particular, although VFGL requires a fixed parametric form, DVFN is slightly better than VFGL. Overall, we can say that DVFN can achieve quality solution with reasonable computation time without specifying any parametric form in advance. Time elapsed per iteration scaled by first iteration time are presented in Figure 4a. The computation time of SDDP increases almost linearly with the number of iterations. However, DVFN exhibits almost constant computation time over all iterations. Similar results can be obtained with different parameters and more number of stages, which are presented in Appendix C.1.

4.2 Energy planning

We consider a 15-stage energy planning problem with 2 scenario branches, in which the objective is to determine optimal hydro and thermal electricity generation levels to satisfy deterministic demand while minimizing the sum of expected cost for electricity production and the penalty from the reservoir level. This example is a slightly simplified version of Guigues (2014). The formulation and details of decision variables and parameters are in Appendix B.2.

In VFGL, the following parametric form was used to capture the cost reduction by increasing the portion of hydro generation and the decreased disutility from the increased reservoir level.

$$\hat{V}_t(r_{t-1}^{final}, \theta_t = (\theta_t^1, \theta_t^2)) = \theta_t^1 r_{t-1}^{final} + e^{-\theta_t^2 r_{t-1}^{final} + b_t}$$

where $\theta_t^1, \theta_t^2 \in \mathbf{R}$. In this paper, the solutions from VFGL with following other parametric forms are also compared.

$$\hat{V}_t^{quad}(r_{t-1}^{final}, \theta_t = (\theta_t^1, \theta_t^2)) = \theta_t^1 r_{t-1}^{final} + \theta_t^2 (r_{t-1}^{final})^2$$
$$\hat{V}_t^{linear}(r_{t-1}^{final}, \theta_t = (\theta_t^1, \theta_t^2)) = \theta_t^1 r_{t-1}^{final}$$

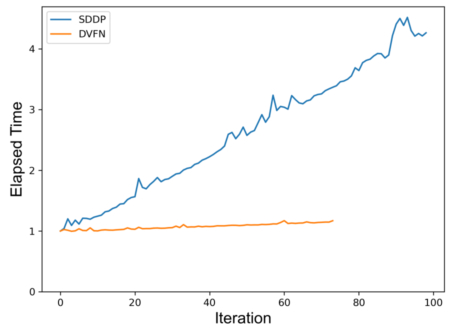
Experiment results are presented in Figure 3 and Table 4 in the same way of the previous example. Here, we can make similar observations as well. For this example, VFGL is showing worse performance in terms of both objective value and first stage solutions compared to SDDP and DVFN regardless of the chosen parametric form. VFGL requires smallest computation time, but this would be less meaningful since the objective value and first stage solutions are worse than the others. Between SDDP and DVFN, we can see that DVFN is performing much better than SDDP, because it shows smaller standard error for the objective and much smaller computation time. Also, Figure 3 shows that DVFN converges to the optimal solution in a very stable way. Time elapsed per iteration are presented in Figure 4b, where DVFN maintains almost constant computation time per iteration just like the previous example. Again, similar results can be obtained with different parameters and more number of stages, which are presented in Appendix C.2.

5 CONCLUSION

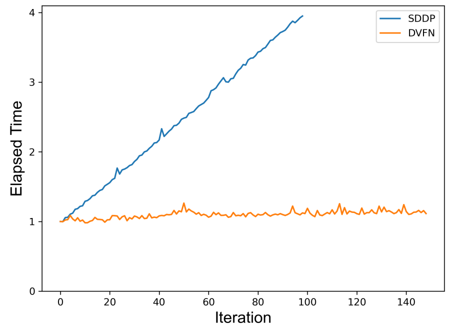
A neural-networks based novel stagewise decomposition algorithm, DVFN, was proposed to solve large-scale multistage stochastic programs efficiently. DVFN approximates the value function using input convex neural networks (ICNN) and finds the weights of ICNN by closely matching the gradient (with respect to inputs) of ICNN to that of the true value function. Numerical experiments showed that DVFN can find optimal solutions of large-scale MSP problems in a stable way. The performance of DVFN was better than SDDP and almost the same as VFGL with the best parametric choice. Hence, DVFN can achieve almost the state-of-the-art performance without the need to specify parametric form of value functions in advance. The art of choosing appropriate parametric form became a simple labor of hyperparameter search for neural networks. Experiments with perturbed problems in Appendix C also show that DVFN can be very useful for real-world applications where a lot of similar problems with slightly different problem settings should be solved (Bertsimas and Stellato, 2021). Still, there is room for improvement for DVFN in terms of standard errors of objective and first stage solutions and computation time. We believe that these can be much resolved if we can make the primal-dual interior point method with ICNN more efficient.

Table 2: Performance comparisons of algorithms for energy planning

Algorithm	Objective	Hydro	Thermal	Time (s)
MSP	769	3.85	16.15	1132
SDDP	769 (1.89)	3.86 (0.02)	16.14 (0.02)	1115 (2.43)
VFGLexp	783 (2.97)	0.00 (0.00)	20.00 (0.00)	206 (1.43)
VFGLquad	1168 (2.22)	16.45 (0.23)	3.55 (0.23)	190 (1.02)
VFGLlinear	776 (2.24)	0.00 (0.00)	20.00 (0.00)	149 (0.14)
DVFN	769 (1.70)	3.84 (0.03)	16.16 (0.03)	519 (1.14)



(a) Production Optimization



(b) Energy Planning

Figure 4: Time elapsed per iteration

Acknowledgements

This research was supported by the National Research Foundation of Korea (NRF) grant funded by the Ministry of Science and ICT (NRF-2020R1A2C101067713 and NRF-2022M3J6A1063021) and the Ministry of Education (NRF-2022R1I1A4069163).

Reference

Amos, B., Xu, L., and Kolter, J. Z. (2017). Input convex neural networks. In *International Conference on Machine Learning*, pages 146–155. PMLR.

Berg, B. P., Denton, B. T., Erdogan, S. A., Rohleder, T., and Huschka, T. (2014). Optimal booking and scheduling in outpatient procedure centers. *Computers & Operations Research*, 50:24–37.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2).

Bertsimas, D. and Stellato, B. (2021). The voice of optimization. *Machine Learning*, 110(2):249–277.

Boyd, S., Boyd, S. P., and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Carino, D. R., Kent, T., Myers, D. H., Stacy, C., Sylvanus, M., Turner, A. L., Watanabe, K., and Ziemba, W. T. (1994). The russell-yasuda kasai model: An asset/liability model for a japanese insurance company using multistage stochastic programming. *Interfaces*, 24(1):29–49.

Carøe, C. C. and Schultz, R. (1999). Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24(1-2):37–45.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.

Czarnecki, W. M., Osindero, S., Jaderberg, M., Swirszcz, G., and Pascanu, R. (2017). Sobolev training for neural networks. *Advances in Neural Information Processing Systems*, 30.

Dai, H., Xue, Y., Syed, Z., Schuurmans, D., and Dai, B. (2021). Neural stochastic dual dynamic programming. *arXiv preprint arXiv:2112.00874*.

Dumouchelle, J., Patel, R., Khalil, E. B., and Bodur, M. (2022). Neur2sp: Neural two-stage stochastic programming. *arXiv preprint arXiv:2205.12006*.

Fadda, E., Perboli, G., and Tadei, R. (2019). A progressive hedging method for the optimization of social engagement and opportunistic iot problems. *European Journal of Operational Research*, 277(2):643–652.

Fletcher, R. (1987). Practical methods of optimization, a wiley-interscience publication. *Great Britain*.

Fleten, S.-E. and Kristoffersen, T. K. (2008). Short-term hydropower production planning by stochastic programming. *Computers & Operations Research*, 35(8):2656–2671.

Funahashi, K. I. (1989). On the approximate realization of continuous mappings by neural networks. *Neural networks*, 2(3):183–192.

Guigues, V. (2014). Sddp for some interstage dependent risk-averse problems and application to hydro-thermal planning. *Computational Optimization and Applications*, 57(1):167–203.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Karimi, B., Ghomi, S. F., and Wilson, J. (2003). The capacitated lot sizing problem: a review of models and algorithms. *Omega*, 31(5):365–378.

Lee, J., Bae, S., Kim, W. C., and Lee, Y. (2022). Value function gradient learning for large-scale multistage stochastic programming problems. *European Journal of Operational Research*.

Pereira, M. V. and Pinto, L. M. (1991). Multi-stage stochastic optimization applied to energy planning. *Mathematical programming*, 52(1):359–375.

Rockafellar, R. T. and Wets, R. J.-B. (1991). Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of operations research*, 16(1):119–147.

Schalbetter, A. (2020). Input convex neural networks for energy optimization in an occupied apartment. Master's thesis, ETH Zurich.

Shapiro, A., Dentcheva, D., and Ruszczynski, A. (2021). *Lectures on stochastic programming: modeling and theory*. SIAM.

Shapiro, J. F. (1993). Mathematical programming models and methods for production planning and scheduling. *Handbooks in operations research and management science*, 4:371–443.

Shiina, T. and Birge, J. R. (2003). Multistage stochastic programming model for electric power capacity expansion problem. *Japan journal of industrial and applied mathematics*, 20(3):379–397.

Triki, C., Beraldi, P., and Gross, G. (2005). Optimal capacity allocation in multi-auction electricity markets under uncertainty. *Computers & operations research*, 32(2):201–217.

Wagner, H. M. and Whitin, T. M. (1958). Dynamic version of the economic lot size model. *Management science*, 5(1):89–96.

Wright, S. J. (1997). *Primal-dual interior-point methods*. SIAM.

Codes of DVFN, all experiments in the paper, and additional experiments can be found at: `https://github.com/HyunglipBae/DVFN`.

A PROOF OF CONVEXITY OF VALUE FUNCTION

Consider the stage t value function $V_{t+1}(x_t, \xi_{[t]}) = \mathbb{E}_{\cdot|\xi_{[t]}}[\mathcal{V}_{t+1}(x_t, \xi_{[t+1]})]$. Its convexity with respect to x_t can be shown by the convexity of $\mathcal{V}_{t+1}(x_t, \xi_{[t+1]})$, because expectation preserves convexity. We show the convexity of $\mathcal{V}_{t+1}(x_t, \xi_{[t+1]})$ by mathematical induction.

Assume that $\mathcal{V}_{t+1}(x_t, \xi_{[t+1]})$ is convex in x_t.

Then, we us show that $\mathcal{V}_t(x_{t-1}, \xi_{[t]})$ is convex in x_{t-1}. From Equation (3),

$$\mathcal{V}_t(x_{t-1}, \xi_{[t]}) = \inf_{x_t \in \mathcal{X}_t(x_{t-1}, \xi_t)} \{f_t(x_t, \xi_t) + V_{t+1}(x_t, \xi_{[t]})\} = \inf_{x_t} \psi(x_{t-1}, x_t, \xi_{[t]}),$$

where

$$\psi(x_{t-1}, x_t, \xi_{[t]}) = f_t(x_t, \xi_t) + V_{t+1}(x_t, \xi_{[t]}) + \mathbf{I}_{\mathcal{X}_t(x_{t-1}, \xi_t)}(x_t),$$

$$\mathbf{I}_C(\cdot) = \begin{cases} 0 & if \quad \cdot \in C \\ +\infty & otherwise \end{cases}$$

Here, $f_t(x_t, \xi_t)$ is convex in x_t by definition, and $V_{t+1}(x_t, \xi_{[t]})$ is convex by the induction hypothesis. Let us extend the feasible region $\mathcal{X}_t(x_{t-1}, \xi_t)$, which is a set of x_t's given x_{t-1}, into the set of (x_{t-1}, x_t) pair. Then, it is a convex set, because it is an intersection of hyperplanes and sublevel sets of convex functions. Thus, $\mathbf{I}_{\mathcal{X}_t(x_{t-1}, \xi_t)}(x_t)$ is convex in (x_{t-1}, x_t). Since the sum of convex functions is again convex, $\psi(x_t, x_{t-1}, \xi_{[t]})$ is convex in (x_{t-1}, x_t). Therefore, $\mathcal{V}_t(x_{t-1}, \xi_{[t]})$, an infimal projection of $\psi(x_t, x_{t-1}, \xi_{[t]})$ with respect to x_t, is convex in x_{t-1}.

For $t = T$, $V_{t+1} = 0$. Hence, $\mathcal{V}_{T+1}$ is convex in x_T.

By the mathematical induction, $\mathcal{V}_{t+1}(x_t, \xi_{[t+1]})$ is convex with respect to x_t, for $t = T-1, \ldots, 1$

B FORMULATION OF NUMERICAL EXAMPLES

In this section, we present the details of decision variables, parameters, and stagewise subproblem formulations of production optimization and energy planning problems shown in Section 4.

B.1 Production Optimization

The factory can produce, outsource, and store three products ($i = 1, 2, 3$) to meet uncertain demands. The decision variables and parameters are presented in Table 3.

An important assumption is that their is an upper limit to the total resource for production (e.g., human resource, machine operation time) for each stage and remainders cannot be reused in next stage. This assumption is expressed as resource limit constraints.

$$\sum_{i \in I} x_{t,i} a_{t,i} \leq r_t$$

Therefore, at each stage, the factory manager must decide how much to produce, outsource and store each product considering the current realized demand and future uncertain demand. This is reflected as the storage balance constraints.

$$s_{t,i} = s_{t-1,i} + x_{t,i} + y_{t,i} - d_{t,i}$$

Any product remaining after satisfying the demand is stored for later stages, which incurs storage cost $s_{t,i}c_{t,i}$. When the demands cannot be met with current storage and production, the factory manager can outsource the shortage at relatively high cost $y_{t,i}b_{t,i}$. The objective at stage t is to minimize the total cost considering current and future stage which is expressed as follows.

$$\sum_{i \in I} y_{t,i} b_{t,i} + \sum_{i \in I} s_{t,i} c_{t,i} + V_{t+1}(s_t)$$

Table 3: Variables for production optimization problem

		Description	Value
Decision variables	$x_{t,i}$	Product i produced at stage t	-
	$y_{t,i}$	Product i outsourced at stage t	-
	$s_{t,i}$	Product i stored at the end of stage t	-
Parameters	$a_{t,i}$	Production cost of product i at stage t	$(1, 2, 5)$
	$b_{t,i}$	Outsourcing cost of product i at stage t	$(6, 12, 20)$
	$c_{t,i}$	Storage cost of product i from the end of stage t to beginning of stage t	$(3, 7, 10)$
	r_t	Maximum production resource available at stage t	10
	$d_{t,i}$	Random demand of product i at stage t	$\{(5, 3, 1), (6, 2, 1), (1, 2, 2)\}$

As shown in Table 3, all parameters $a_{t,i}, b_{t,i}, c_{t,i}, r_t, d_{t,i}$ are assumed to have identical values or distributions in all stages for simplicity. $(d_{t,1}, d_{t,2}, d_{t,3})$ takes one of three values given in the table with equal probability.

Formulations of stagewise sumproblems are given below.

- Stage 1 subproblem

$$\begin{aligned} \text{minimize} \quad & \sum_{i \in I} y_{1,i} b_{1,i} + \sum_{i \in I} s_{1,i} c_{1,i} + V_2(s_1) & & \\ \text{subject to} \quad & \sum_{i \in I} x_{1,i} a_{1,i} \le r_1 & \forall i \in I \quad & \text{Resource limit} \\ & s_{1,i} = x_{1,i} + y_{1,i} & \forall i \in I \quad & \text{Storage balance} \\ & x_{1,i}, y_{1,i}, s_{1,i} \ge 0 & \forall i \in I \quad & \text{Non-negativity} \end{aligned}$$

- Stage t subproblem ($t = 2, \ldots, T-1$)

$$\begin{aligned} \text{minimize} \quad & \sum_{i \in I} y_{t,i} b_{t,i} + \sum_{i \in I} s_{t,i} c_{t,i} + V_{t+1}(s_t) & & \\ \text{subject to} \quad & \sum_{i \in I} x_{t,i} a_{t,i} \le r_t & \forall i \in I \quad & \text{Resource limit} \\ & s_{t,i} = s_{t-1,i} + x_{t,i} + y_{t,i} - d_{t,i} & \forall i \in I \quad & \text{Storage balance} \\ & x_{t,i}, y_{t,i}, s_{t,i} \ge 0 & \forall i \in I \quad & \text{Non-negativity} \end{aligned}$$

- Stage T subproblem

$$\begin{aligned} \text{minimize} \quad & \sum_{i \in I} y_{T,i} b_{T,i} & & \\ \text{subject to} \quad & \sum_{i \in I} x_{T,i} a_{T,i} \le r_T & \forall i \in I \quad & \text{Resource limit} \\ & s_{T,i} = s_{T-1,i} + x_{T,i} + y_{T,i} - d_{T,i} & \forall i \in I \quad & \text{Storage balance} \\ & x_{T,i}, y_{T,i}, s_{T,i} \ge 0 & \forall i \in I \quad & \text{Non-negativity} \end{aligned}$$

B.2 Energy planning

In this problem, there are hydro plants and thermal plants for electricity generation. The decision variables and parameters are presented in Table 4

Hydro power generation requires a low cost, but resources are limited by the size of reservoirs where the amount of water inflow to reservoirs is uncertain. The initial water reservoir level at the beginning of stage t can be expressed as the final water reservoir level at the end of stage $t-1$ plus the uncertain water inflow between stages $t-1$ and t as follows.

$$r_t^{init} = r_{t-1}^{final} + I_t$$

During stage t, W_t is used for power generation from the reservoir r_t^{init}. Of course, the final reservoir level should be non-negative.

$$r_t^{final} = r_t^{init} - W_t$$

$$r_t^{final} \geq 0$$

On the other hand, the thermal power generation has no resource limits but requires a higher cost. Of course, the amount of power generation from hydro and thermal plants should satisfy the demand.

$$W_t + H_t \geq d_t$$

The objective of each subproblem is to minimize total cost associated with power generation. At stage t, we set the objective as follows.

$$c_t^W W_t + c_t^H H_t + e^{-a_t r_t^{final} + b_t} + V_{t+1}(r_t^{final})$$

Here, $c_t^W W_t + c_t^H H_t$ is the cost of hydro and thermal power generation, $e^{-a_t r_t^{final} + b_t}$ is the environmental disutility from the remaining reservoir level, and $V_{t+1}(r_t^{final})$ is the value function for later stages. As shown in Table 4, all parameters $c_t^W, c_t^H, d_t, a_t, b_t, I_t$ are assumed to have identical values or distributions in all stages for simplicity. The water inflow to reservoir in each stage is independent and identically distributed, which follows a normal distribution with mean 20 and standard deviation 5. The stagewise sumproblems are as follows

- Stage 1 subproblem

$$\begin{array}{lll}
\text{minimize} & c_1^W W_1 + c_1^H H_1 + e^{-a_1 r_1^{final} + b_1} + V_2(r_1^{final}) & \\
\text{subject to} & r_1^{init} = r_0^{init} & \text{Initial reservoir} \\
& r_1^{final} = r_1^{init} - W_1 & \text{Reservoir balance} \\
& W_1 + H_1 \geq d_1 & \text{Demand} \\
& r_1^{final}, W_1, H_1 \geq 0 & \text{Non-negativity}
\end{array}$$

- Stage t subproblem ($t = 2, \ldots, T-1$)

$$\begin{array}{lll}
\text{minimize} & c_t^W W_t + c_t^H H_t + e^{-a_t r_t^{final} + b_t} + V_{t+1}(r_t^{final}) & \\
\text{subject to} & r_t^{init} = r_{t-1}^{final} + I_t & \text{Initial reservoir} \\
& r_t^{final} = r_t^{init} - W_t & \text{Reservoir balance} \\
& W_t + H_t \geq d_t & \text{Demand} \\
& r_t^{final}, W_t, H_t \geq 0 & \text{Non-negativity}
\end{array}$$

- Stage T subproblem

$$
\begin{array}{lll}
\text{minimize} & c_T^W W_T + c_T^H H_T + e^{-a_T r_T^{final} + b_T} & \\
\text{subject to} & r_T^{init} = r_{T-1}^{final} + I_T & \text{Initial reservoir} \\
& r_T^{final} = r_T^{init} - W_T & \text{Reservoir balance} \\
& W_T + H_T \geq d_T & \text{Demand} \\
& r_T^{final}, W_T, H_T \geq 0 & \text{Non-negativity}
\end{array}
$$

Table 4: Variables for energy planning problem

		Description	Value
Decision variables	r_t^{init}	Water reservoir level at the beginning of stage t	-
	r_t^{final}	Water reservoir level at the end of stage t	-
	W_t	Hydro electricity generation level at stage t	-
	H_t	Thermal electricity generation level at stage t	-
Parameters	r_0^{init}	Initial water reservoir level	40
	c_t^W	Cost of hydro electricity production per unit at stage t	2
	c_t^H	Cost of thermal electricity production per unit at stage t	7
	d_t	Electricity demand at stage t	20
	a_t	Reservoir level utility coefficient	0.1
	b_t	Reservoir level utility scaling constant	5
	I_t	Water inflow to reservoir at the beginning of stage t	Normally distributed with mean 20, standard deviation 5

C PERTURBED PROBLEMS

In this section, we present the results from perturbed problems to confirm that the observations shown in the main text are not limited to those specific settings. More specifically, we compare the first stage solutions of perturbed problems obtained by SDDP, VFGL and DVFN, where VFGL uses three different parametric forms as in the main text. For production optimization problem, we perturbed the maximum resource r_t, outsourcing cost for product 2 $b_{t,2}$ and the number of stages. For energy planning, the cost of hydro electricity c_t^W, thermal electricity c_t^H, reservoir level utility coefficient a_t and reservoir level utility scaling constant b_t are perturbed. The results are summarized in Tables 5, 6 for production optimization and Tables 7, 8 for energy planning and they clearly show that the performance of VFGL is unstable depending on the chosen parametric form and especially in the energy planning problem, depending on the perturbed parameters.

Furthermore, we performed additional experiments with larger number of stages than the ones considered in the main text. Note that 11-stage production optimization with 3 scenario branches and 15-stage energy planning with 2 scenario branches are considered in the main text, because these are the maximum problem size that MSP could handle with our PC with 64GB RAM. Here, we consider 12, 13, 14-stage production optimization problems and 15, 16, 17-stage energy planning problems. For these problems, benchmark solutions from MSP cannot be obtained. Therefore, we run each algorithm (SDDP, VFGL, DVFN) until the change in the objective value in the recent three iterations is less than 0.02%. The results are summarized in Table 10 for production optimization and Table 9 for energy planning. For both problems, DVFN shows almost same objective value with less computational time. VFGL is faster than others, but again, it is very sensitive to predetermined parametric form. We can see that DVFN finds accurate solutions efficiently without worrying about parametric forms.

Table 5: Comparison of perturbed problems of r_t with MSP, SDDP, VFGLexp, VFGLquad, VFGLlinear and DVFN for production optimization problem

	MSP		SDDP		VFGLexp		VFGLquad		VFGLlinear		DVFN	
r_t	$x_{1,1}, x_{1,2}, x_{1,3}$	Obj	$x_{1,1}, x_{1,2}, x_{1,3}$	obj	$x_{1,1}, x_{1,2}, x_{1,3}$	obj	$x_{1,1}, x_{1,2}, x_{1,3}$	obj	$x_{1,1}, x_{1,2}, x_{1,3}$	obj	$x_{1,1}, x_{1,2}, x_{1,3}$	obj
8.0	2.00, 0.00, 1.20	312	2.00, 0.00, 1.20	312	1.74, 0.24, 1.13	312	3.17, 0.00, 0.00	318	0.00, 0.00, 0.00	323	2.02, 0.20, 1.12	312
8.5	2.50, 0.00, 1.19	286	2.50, 0.00, 1.17	286	1.77, 0.00, 1.15	286	2.51, 0.00, 0.00	293	0.00, 0.00, 0.00	297	2.49, 0.00, 1.20	286
9.0	2.00, 0.00, 1.36	260	2.00, 0.00, 1.03	260	1.65, 0.00, 1.13	260	2.08, 0.00, 0.00	268	0.00, 0.00, 0.00	271	1.98, 0.00, 1.37	262
9.5	1.50, 0.00, 1.41	235	2.50, 0.00, 1.02	235	1.49, 0.00, 1.03	235	2.06, 0.00, 0.00	242	0.00, 0.00, 0.00	245	1.51, 0.00, 1.51	235
10.5	0.50, 0.00, 1.46	187	0.50, 0.00, 1.06	187	0.48, 0.00, 1.01	187	1.51, 0.00, 0.00	195	0.00, 0.00, 0.00	196	0.50, 0.00, 1.01	187
11.0	0.00, 0.00, 1.44	165	2.00, 0.00, 1.10	165	0.00, 0.00, 1.04	165	1.09, 0.00, 0.00	172	0.00, 0.00, 0.00	173	0.00, 0.00, 1.00	165
11.5	0.00, 0.00, 1.27	146	0.00, 0.00, 1.37	146	0.00, 0.00, 0.95	146	1.00, 0.00, 0.00	152	0.00, 0.00, 0.00	153	0.00, 0.00, 0.90	146
12.0	0.00, 0.00, 1.10	127	0.00, 0.00, 0.89	127	0.00, 0.00, 0.81	127	1.01, 0.00, 0.00	132	0.00, 0.00, 0.00	133	0.00, 0.00, 0.80	127
12.5	0.00, 0.00, 0.92	108	0.00, 0.00, 0.87	108	0.00, 0.00, 0.70	108	1.01, 0.00, 0.00	112	0.00, 0.00, 0.00	113	0.00, 0.00, 0.70	109
13.0	0.00, 0.00, 0.75	89	0.00, 0.00, 0.65	89	0.00, 0.00, 0.60	89	1.00, 0.00, 0.00	92	0.00, 0.00, 0.00	93	0.00, 0.00, 0.60	89
13.5	0.00, 0.00, 0.57	70	0.00, 0.00, 0.52	70	0.00, 0.00, 0.50	70	1.00, 0.00, 0.00	72	0.00, 0.00, 0.00	73	0.00, 0.00, 0.50	70
14.0	0.00, 0.00, 0.40	51	0.00, 0.00, 0.40	51	0.00, 0.00, 0.39	51	0.98, 0.00, 0.00	52	0.00, 0.00, 0.00	53	0.00, 0.00, 0.40	52
14.5	0.00, 0.00, 0.20	32	0.00, 0.00, 0.20	32	0.00, 0.00, 0.20	32	0.49, 0.00, 0.00	33	0.00, 0.00, 0.00	33	0.00, 0.00, 0.23	33
15.0	0.00, 0.00, 0.00	13	0.00, 0.00, 0.00	13	0.00, 0.00, 0.00	13	0.00, 0.00, 0.00	13	0.00, 0.00, 0.00	13	0.00, 0.00, 0.00	14

Table 6: Comparison of perturbed problems of $b_{t,2}$ with MSP, SDDP, VFGLexp, VFGLquad, VFGLlinear and DVFN for production optimization problem

	MSP		SDDP		VFGLexp		VFGLquad		VFGLlinear		DVFN	
$b_{t,2}$	$x_{1,1}, x_{1,2}, x_{1,3}$	Obj	$x_{1,1}, x_{1,2}, x_{1,3}$	obj	$x_{1,1}, x_{1,2}, x_{1,3}$	obj	$x_{1,1}, x_{1,2}, x_{1,3}$	obj	$x_{1,1}, x_{1,2}, x_{1,3}$	obj	$x_{1,1}, x_{1,2}, x_{1,3}$	obj
7.0	0.00, 0.00, 1.00	182	0.00, 0.00, 1.00	182	0.00, 0.00, 1.00	182	1.00, 0.00, 0.00	189	0.00, 0.00, 0.00	190	0.00, 0.00, 1.01	182
8.0	0.00, 0.00, 1.45	203	0.00, 0.00, 1.10	204	0.00, 0.00, 1.09	203	1.00, 0.00, 0.00	212	0.00, 0.00, 0.00	213	0.00, 0.00, 1.33	204
9.0	0.00, 0.00, 1.59	205	0.00, 0.00, 1.10	205	0.00, 0.00, 1.11	205	1.13, 0.00, 0.00	214	0.00, 0.00, 0.00	215	0.01, 0.00, 1.50	206
10.0	0.49, 0.00, 1.53	207	0.70, 0.00, 1.10	207	0.03, 0.00, 1.09	207	1.28, 0.00, 0.00	215	0.00, 0.00, 0.00	232	0.56, 0.00, 1.38	207
11.0	1.00, 0.00, 1.47	208	1.00, 0.00, 1.00	208	0.89, 0.00, 1.02	208	1.50, 0.00, 0.00	216	0.00, 0.00, 0.00	232	1.00, 0.00, 1.43	208
13.0	1.00, 0.00, 1.43	210	1.00, 0.00, 1.10	210	0.99, 0.00, 1.00	210	1.85, 0.00, 0.00	217	0.00, 0.00, 0.00	219	1.01, 0.00, 1.43	210
14.0	1.00, 0.00, 1.52	210	1.00, 0.00, 1.00	210	1.00, 0.00, 1.04	210	1.79, 0.00, 0.00	217	0.00, 0.00, 0.00	219	1.00, 0.00, 1.52	210
15.0	1.00, 0.00, 1.52	210	1.00, 0.00, 1.00	210	0.98, 0.00, 1.00	210	1.91, 0.00, 0.00	217	0.00, 0.00, 0.00	219	1.01, 0.00, 1.54	210
16.0	1.00, 0.00, 1.52	210	1.00, 0.00, 1.10	210	0.99, 0.00, 1.00	210	1.92, 0.00, 0.00	217	0.00, 0.00, 0.00	219	0.98, 0.00, 1.43	210
17.0	1.00, 0.00, 1.51	210	1.00, 0.00, 1.10	210	1.00, 0.00, 1.12	210	1.91, 0.00, 0.00	217	0.00, 0.00, 0.00	234	0.98, 0.00, 1.52	210
18.0	1.00, 0.00, 1.42	210	1.00, 0.00, 1.10	210	0.98, 0.00, 1.02	210	1.77, 0.00, 0.00	217	0.00, 0.00, 0.00	219	1.01, 0.00, 1.52	210
19.0	1.00, 0.00, 1.38	210	1.00, 0.00, 1.10	210	1.00, 0.00, 1.09	210	1.86, 0.00, 0.00	217	0.00, 0.00, 0.00	219	0.95, 0.00, 1.50	210

Table 7: Comparison of perturbed problems of c_t^W, c_t^H with MSP, SDDP, VFGLexp, VFGLquad, VFGLlinear and DVFN for energy planning problem

		MSP		SDDP		VFGLexp		VFGLquad		VFGLlinear		DVFN	
c_t^W	c_t^H	W_1, H_1	Obj	W_1, H_1	obj	W_1, H_1	obj	W_1, H_1	obj	W_1, H_1	obj	W_1, H_1	obj
2.0	5.0	0.00, 20.00	717	0.00, 20.00	721	0.00, 20.00	717	14.58, 5.42	1073	0.00, 20.00	727	0.01, 19.99	718
2.0	5.5	0.29, 19.71	731	0.28, 19.72	731	4.49, 15.51	762	16.14, 3.86	1113	3.89, 16.11	762	0.25, 19.75	731
2.0	6.0	1.62, 18.38	744	1.61, 18.39	745	0.00, 20.00	769	15.06, 4.94	1138	0.00, 20.00	748	1.51, 18.49	746
2.0	6.5	2.8, 17.2	757	2.79, 17.21	767	0.00, 20.00	767	14.50, 5.50	1135	0.00, 20.00	769	3.22, 16.78	757
2.0	7.5	4.81, 15.19	792	4.90, 15.10	792	1.10, 18.90	793	16.63, 3.37	1274	1.75, 18.25	792	4.68, 15.32	792
2.0	8.0	5.68, 14.32	1044	5.67, 14.33	1044	5.42, 14.58	1044	16.18, 3.82	1332	5.44, 14.56	1044	5.59, 14.41	1044
3.0	7.0	1.62, 18.38	1081	1.68, 18.32	1081	0.00, 20.00	1081	16.73, 3.27	1147	5.10, 14.90	1081	3.51, 16.49	1081
3.5	7.0	0.29, 19.71	1181	0.33, 19.67	1184	3.43, 16.57	1187	20.00, 0.00	1794	0.00, 20.00	1205	0.27, 19.73	1181
4.0	7.0	0.00, 20.00	1317	0.00, 20.00	1317	0.00, 20.00	1324	20.00, 0.00	1861	0.00, 20.00	1323	0.02, 19.98	1322
4.5	7.0	0.00, 20.00	1452	0.00, 20.00	1454	0.00, 20.00	1469	20.00, 0.00	1862	0.00, 20.00	1481	0.00, 20.00	1452
5.0	7.0	0.00, 20.00	1587	0.00, 20.00	1588	0.00, 20.00	1594	20.00, 0.00	1909	0.00, 20.00	1593	0.00, 20.00	1587

Table 8: Comparison of perturbed problems of a_t, b_t with MSP, SDDP, VFGLexp, VFGLquad, VFGLlinear and DVFN for energy planning problem

		MSP		SDDP		VFGLexp		VFGLquad		VFGLlinear		DVFN	
a_t	b_t	W_1, H_1	Obj	W_1, H_1	obj	W_1, H_1	obj	W_1, H_1	obj	W_1, H_1	obj	W_1, H_1	obj
0.1	3.0	20.00, 0.00	670	20.00, 0.00	670	20.00, 0.00	671	20.00, 0.00	696	20.00, 0.00	670	20.00, 0.00	670
0.1	3.5	18.85, 1.15	694	18.9, 1.10	697	20.00, 0.00	724	20.00, 0.00	717	19.17, 0.83	695	19.98, 0.02	699
0.1	4.0	13.85, 6.15	719	13.81, 6.19	719	8.53, 11.47	719	18.28, 1.72	734	7.35, 12.65	726	15.63, 4.37	719
0.1	4.5	8.85, 11.15	744	8.96, 11.04	744	1.67, 18.33	753	10.90, 9.10	750	1.78, 18.22	759	8.84, 11.16	749
0.1	5.5	0.00, 20.00	794	0.00, 20.00	797	0.00, 20.00	810	0.00, 20.00	794	0.00, 20.00	814	0.00, 20.00	794
0.1	6.0	0.00, 20.00	821	0.00, 20.00	826	0.00, 20.00	821	0.00, 20.00	832	0.00, 20.00	838	0.00, 20.00	826
0.2	5.0	19.42, 0.58	677	19.42, 0.58	677	17.91, 2.09	690	20.00, 0.00	687	17.68, 2.32	683	20.00, 0.00	687
0.25	5.0	20.00, 0.00	658	20.00, 0.00	658	20.00, 0.00	673	20.00, 0.00	686	20.00, 0.00	692	20.00, 0.00	658
0.3	5.0	20.00, 0.00	647	20.00, 0.00	650	20.00, 0.00	662	20.00, 0.00	668	20.00, 0.00	661	20.00, 0.00	647
0.35	5.0	20.00, 0.00	640	20.00, 0.00	640	20.00, 0.00	648	20.00, 0.00	677	20.00, 0.00	640	20.00, 0.00	640
0.4	5.0	20.00, 0.00	635	20.00, 0.00	635	20.00, 0.00	643	20.00, 0.00	665	20.00, 0.00	635	20.00, 0.00	635

Table 9: Comparison of problems with larger number of stages with SDDP, VFGLexp, VFGLquad, VFGLlinear and DVFN for production optimization problem

	SDDP			VFGLexp			VFGLquad			VFGLlinear			DVFN		
T	$x_{1,1}.x_{1,2},x_{1,3}$	obj	time	$x_{1,1}.x_{1,2},x_{1,3}$	obj	time	$x_{1,1}.x_{1,2},x_{1,3}$	obj	time	$x_{1,1}.x_{1,2},x_{1,3}$	obj	time	$x_{1,1}.x_{1,2},x_{1,3}$	obj	time
12	1.00, 0.00, 1.00	230	650	0.97, 0.00, 1.23	231	434	1.65, 0.00, 0.00	239	432	0.00, 0.00, 0.00	241	302	1.01, 0.00, 1.48	231	502
13	1.00, 0.00, 1.07	251	711	0.99, 0.00, 1.26	251	474	2.01, 0.00, 0.00	260	472	0.00, 0.00, 0.00	264	329	1.01, 0.00, 1.47	252	553
14	1.00, 0.00, 1.07	275	714	0.96, 0.00, 1.04	275	514	1.81, 0.00, 0.00	283	512	0.00, 0.00, 0.00	285	360	1.00, 0.00, 1.43	275	541

Table 10: Comparison of problems with larger number of stages with SDDP, VFGLexp, VFGLquad, VFGLlinear and DVFN for energy planning problem

	SDDP			VFGLexp			VFGLquad			VFGLlinear			DVFN		
T	W_1, H_1	obj	time	W_1, H_1	obj	time	W_1, H_1	obj	time	W_1, H_1	obj	time	W_1, H_1	obj	time
16	3.28, 16.72	808	1601	0.00, 20.00	813	109	15.28, 4.72	1237	203	0.00, 20.00	823	85	3.20, 16.80	807	417
17	2.82, 17.18	864	1191	0.00, 20.00	857	116	15.48, 4.52	1274	217	0.00, 20.00	867	90	2.84, 17.16	848	369
18	2.37, 17.63	892	1450	0.00, 20.00	895	123	15.19, 4.10	1324	230	0.00, 20.00	903	184	2.79, 17.21	893	367

D HYPERPARAMETER SEARCH

For DVFN, the hyperparameters of FICNNs include the number of nodes in each layer, the number of hidden layers, activation, epochs per iteration, optimizer, and learning rate. There are various methods to tune hyperparameters such as gird search, random search, or line search for example (Bergstra and Bengio, 2012). Grid and random search optimize different parameters simultaneously, whereas each parameter is tuned independently in line search. In this paper, we adopted the line search as in Schalbetter (2020) due to enormous possible combinations of hyperparameters based on median value of objective value and total elapsed time.

Table 11 shows the search range of hyperparameters, and the chosen hyperparameter are indicated with $*$. To see the performance of DVFN depending on the values of hyperparameters, we solved the production and energy planning problems with DVFN for 10 times, and the box plots of objective value and elapsed time are shown in Figure 5, 6, 7 and 8.

Table 11: Candidates of hyperparameters for the FICNNs for production optimization and energy planning problems (* indicates the chosen hyperparameter)

Hyperparameters	Production Optimization	Energy Planning
Epochs per iteration	5*, 10, 15, 20	5*, 10, 15, 20
Number of Nodes	32, 64*, 128, 256	32, 64*, 128, 256
Number of Hidden Layers	1*, 2, 3	1*, 2, 3
Activation	ReLU, ELU, Softplus*	ReLU, ELU*, Softplus
Optimizer	Adam*, Adagrad, SGD, RMSProp	Adam*, Adagrad, SGD, RMSProp
Learning rate	0.0005, 0.001, 0.0015*, 0.002	0.0005, 0.001*, 0.0015*, 0.002

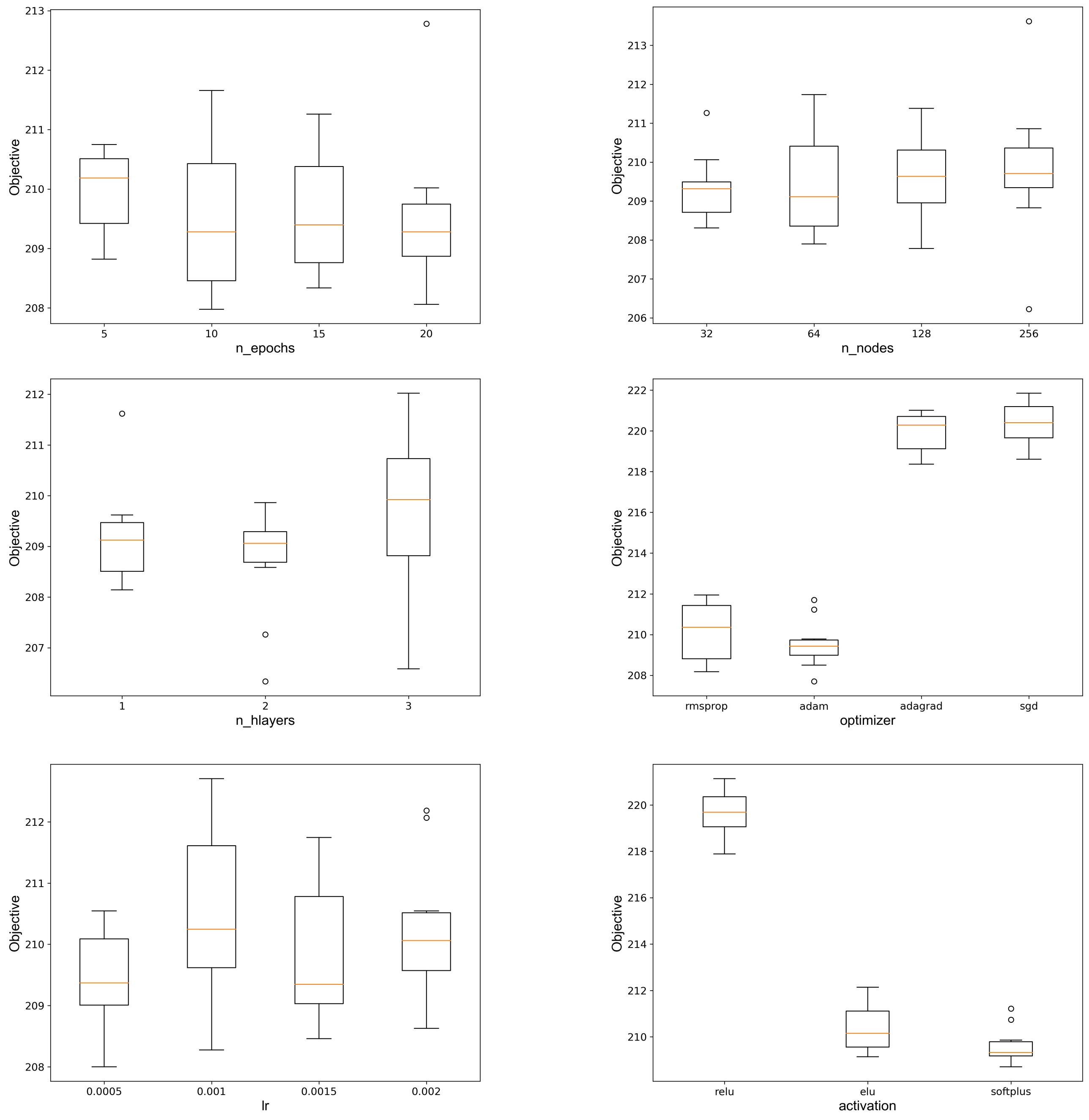


Figure 5: Boxplots of objective value with various hyperparameters for production optimization

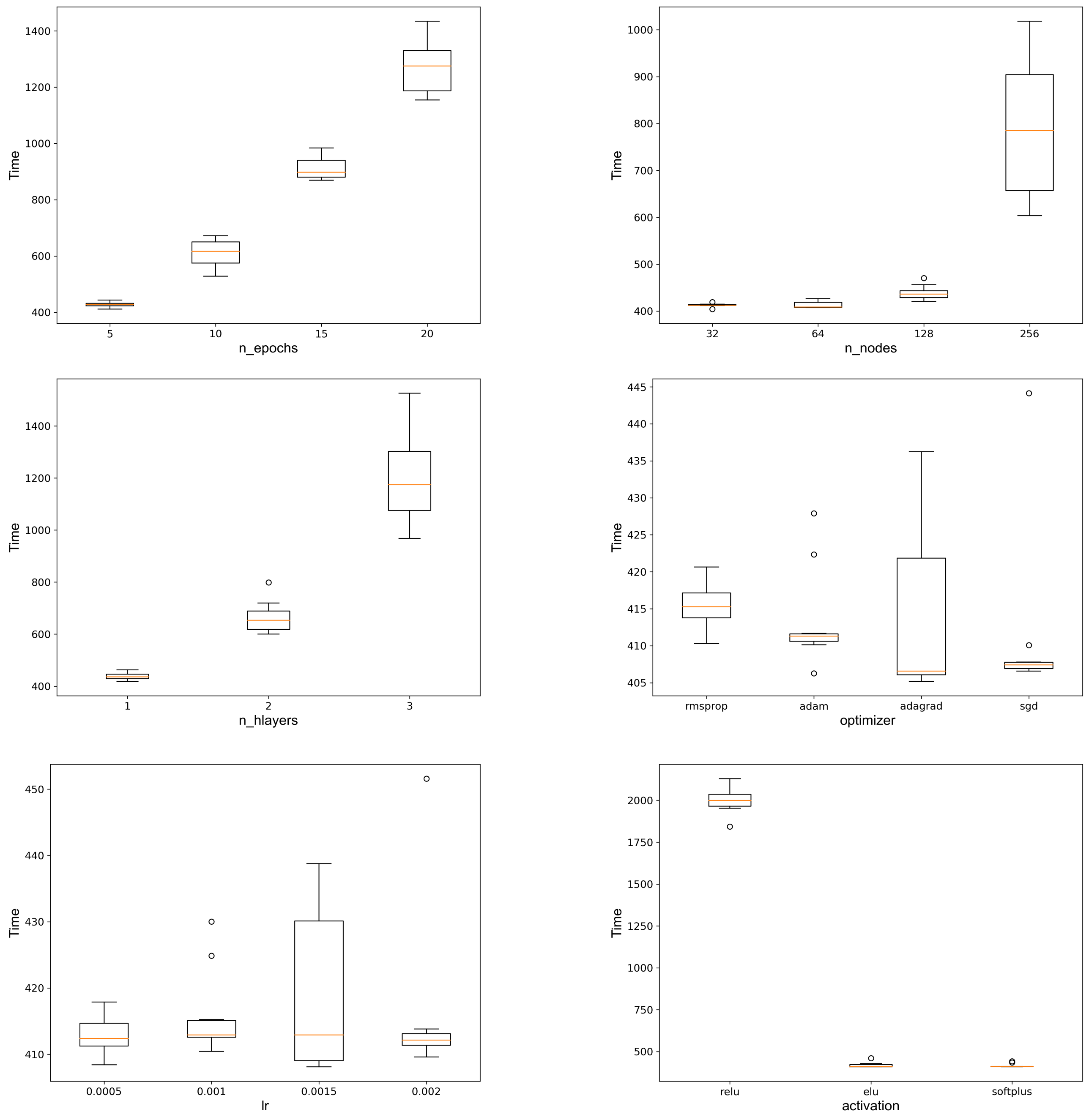


Figure 6: Boxplots of total elapsed time with various hyperparameters for production optimization

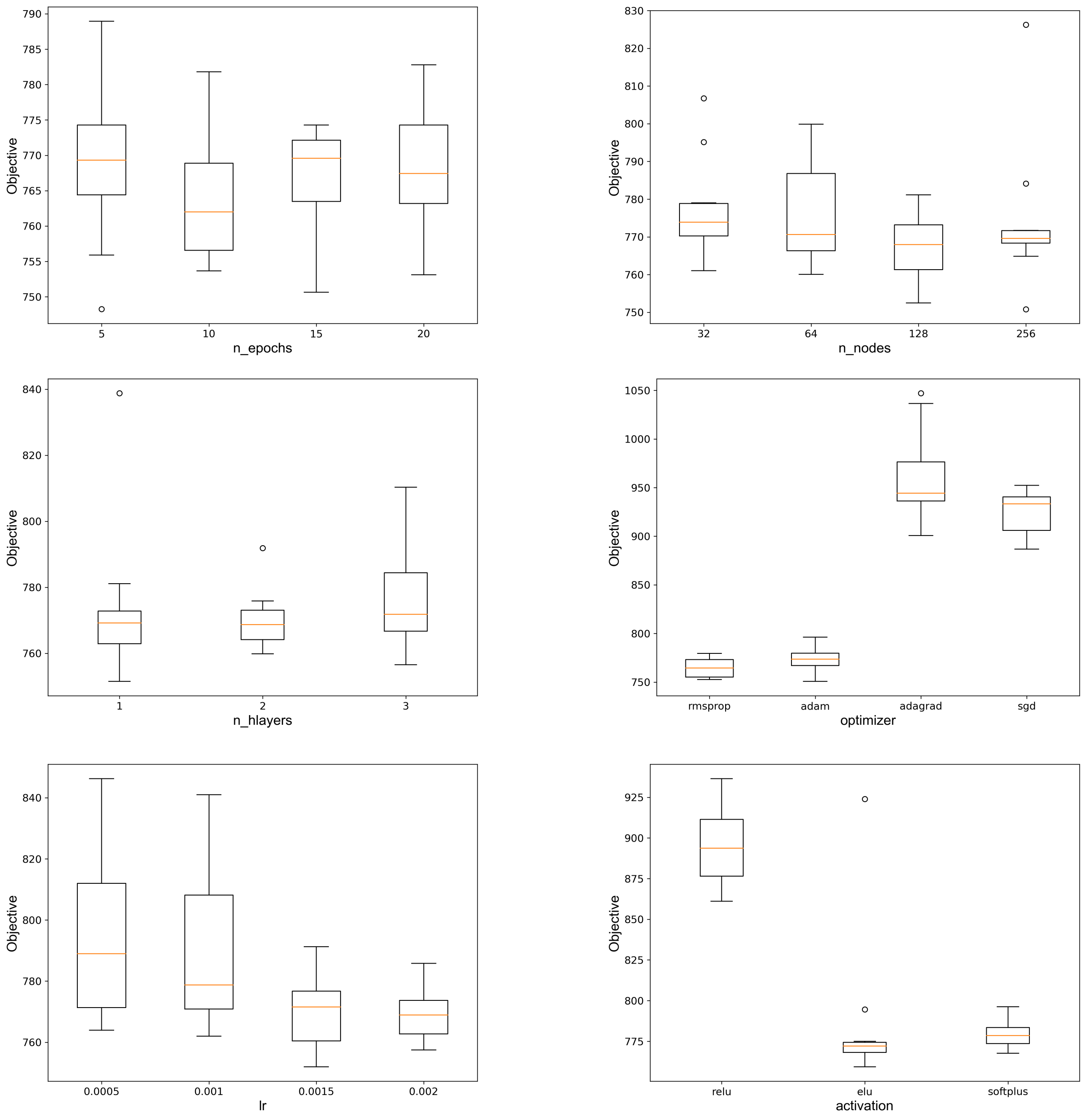


Figure 7: Boxplots of objective value with various hyperparameters for energy planning

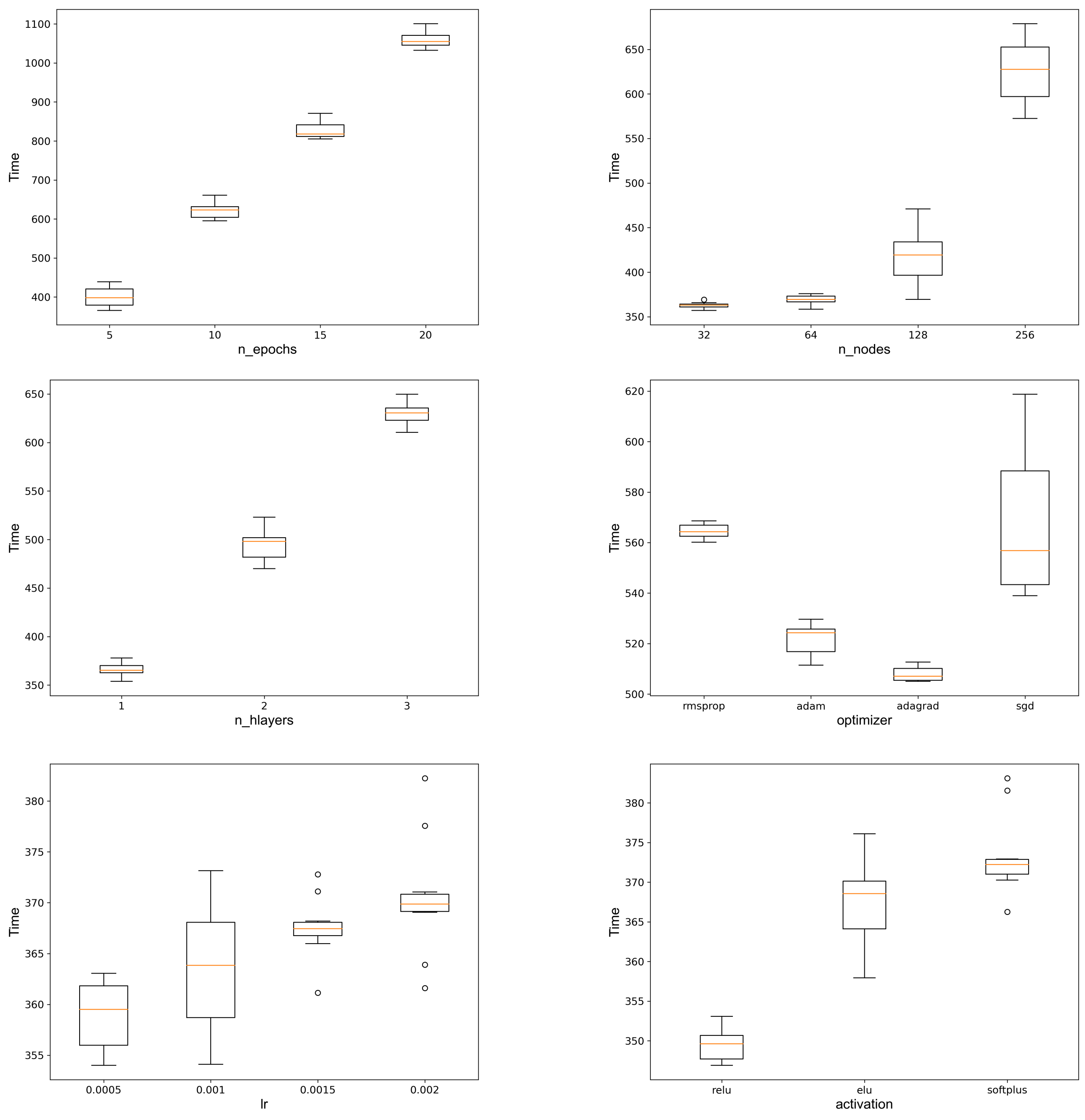
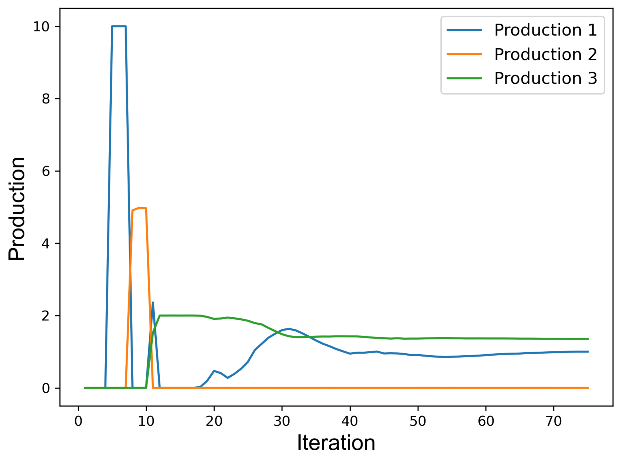


Figure 8: Boxplots of total elapsed time with various hyperparameters for energy planning

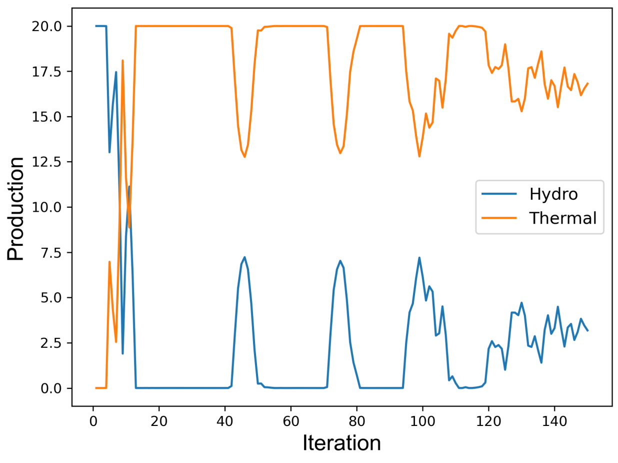
E PRACTICAL BENEFIT OF INPUT CONVEX NEURAL NETWORKS

DVFN approximates the value function with input convex neural networks (ICNN). We have chosen ICNN because it has a clear theoretical advantage that its convexity with respect to inputs guarantees the convergence to the optimal solution as in VFGL (see Section 3). To further demonstrate the practical advantage of using ICNN, we tested DVFN by replacing ICNN with feedforward networks (FFN). The average values and standard errors of objective function value, first stage solutions, and computational time after 20 repeated experiments are represented in Tables 12, 13 and the evolution of the first stage solutions for one instance is depicted in Figure 9.

In production optimization, DVFN (FFN) shows stable convergence to the optimal solution and generates quite accurate solution compared to MSP. However, DVFN (FFN) shows almost double of the computational time compared to DVFN (ICNN). Conversely, in energy planning, DVFN (FFN) shows almost same computational time with DVFN (ICNN). However, DVFN (FFN) shows the first stage solution which is quite different to MSP with large variance and unstable convergence. Both cases clearly indicate that the practical advantage of using ICNN.



(a) Production Optimization



(b) Energy Planning

Figure 9: First stage decisions of DVFN (FFN)

Table 12: Comparison of ICNN and FFN for production optimization

Algorithm	Objective	Production 1	Production 2	Production 3	Time (s)
MSP	210	1.00	0.00	1.50	10215
DVFN (FFN)	210 (0.18)	0.98 (0.00)	0.00 (0.00)	1.37 (0.02)	1086 (46.28)
DVFN (ICNN)	210 (0.14)	0.99 (0.00)	0.00 (0.00)	1.45 (0.01)	574 (1.36)

Table 13: Comparison of ICNN and FFN for energy planning

Algorithm	Objective	Hydro	Thermal	Time (s)
MSP	769	3.85	16.15	1132
DVFN (FFN)	770 (1.18)	2.74 (0.33)	17.26 (0.33)	485 (5.20)
DVFN (ICNN)	769 (1.70)	3.84 (0.03)	16.16 (0.03)	519 (1.14)