Deep Financial Planning

Quantitative Finance Conference
in honor of Michael Dempster's 85th Birthday
Cambridge, UK

April 13, 2023

Woo Chang Kim
Professor
Industrial and Systems Engineering Department, and
Graduate School of Data Science
KAIST
wkim@kaist.ac.kr
http://felab.kaist.ac.kr

Contents

Part 1

MOTIVATION AND PROBLEM DESCRIPTION

Based on:
Woo Chang Kim, Do-Gyun Kwon, Yongjae Lee, Jang Ho Kim & Changle Lin (2020) Personalized goal-based investing via multi-stage stochastic goal programming, *Quantitative Finance*, 20:3, 515-526

Motivation: National Pension Service of Korea

- The National Pension Service of Korea (NPS)
  - Public pension fund in South Korea
  - Third largest in the world with $800 billion in assets
  - Largest investor in South Korea

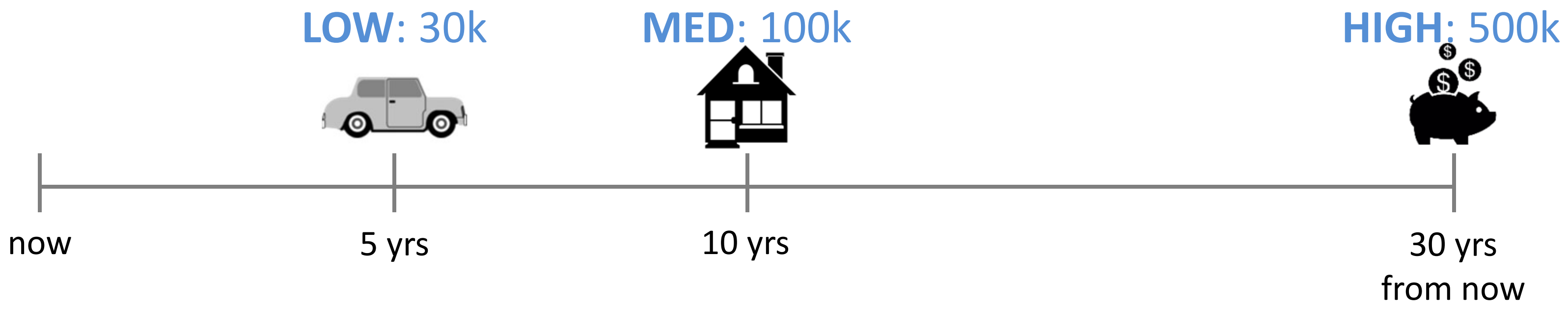
- A while back, CEO of NPS wanted…
  - An ALM system that serves all NPS participants (25 million)
  - Includes not only NPS but also retirement and individual pensions
  - Provides personalized reports to all participants periodically (quarterly, at least)
  - Ideally, allows users to "play with it" so that they can plan things ahead and understand the importance of NPS

Example Individual

- Life time spending goals with priorities
  - *LOW*: Car purchase in 5 years ($30,000)
  - *MEDIUM*: Home purchase in 10 years ($100,000)
  - *HIGH*: Retirement savings in 30 years ($500,000)

- Lower priority goals occur before the more important goals

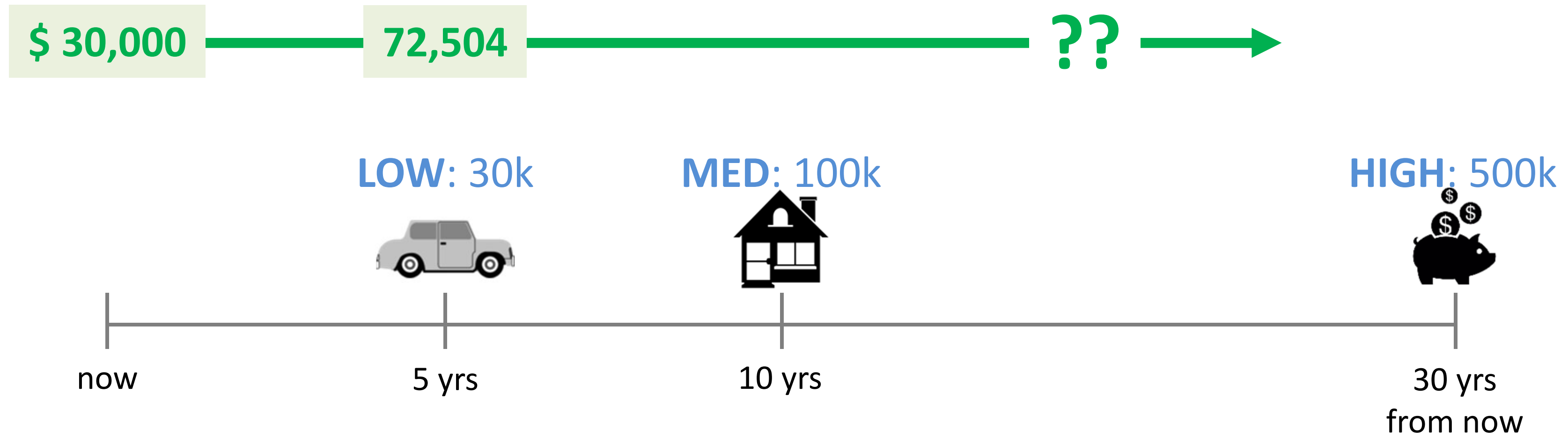
- Then, should individuals spend as goals occur or should they save for future more important goals?



Should You Spend?

- After 5 years, is it optimal to purchase the car?

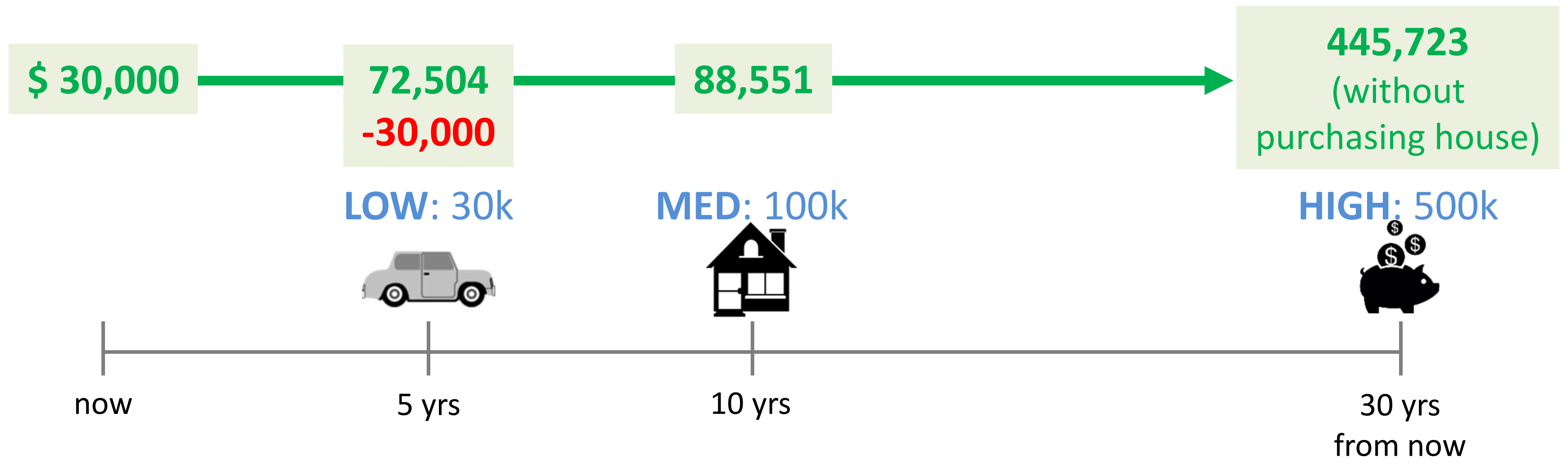
- What would you base your decision on?
  - Need to consider: How does decision now affect the achievement of future more important goals?



Should You Spend?

- It will be helpful if individuals have information on possible outcomes

- If you purchase a $30,000 car, then
  - Other two goals cannot be reached
  - Highest priority cannot be met even when skipping the medium goal

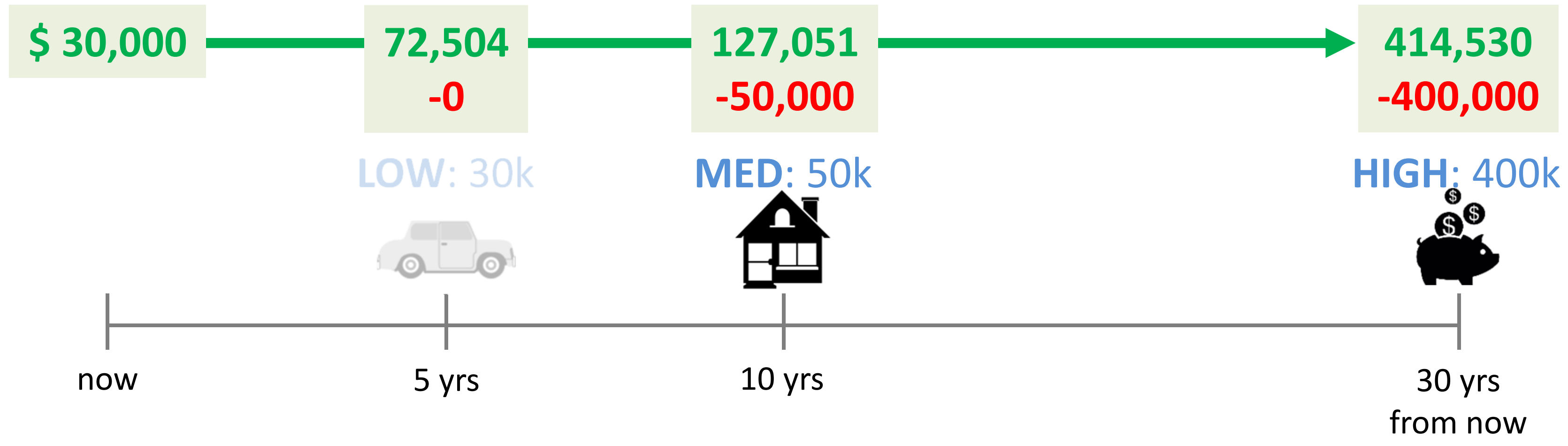


Advice: Modify Goals

- It will be helpful if the individual has information on possible outcomes

- **Advice (1): Reduce goal amounts**
  - Skip purchasing a car
  - Reduce home purchase goal amount to $50,000
  - Reduce retirement savings goal amount to $400,000

  ➔ **With the reduced(more realistic) goals, they can be achieved**

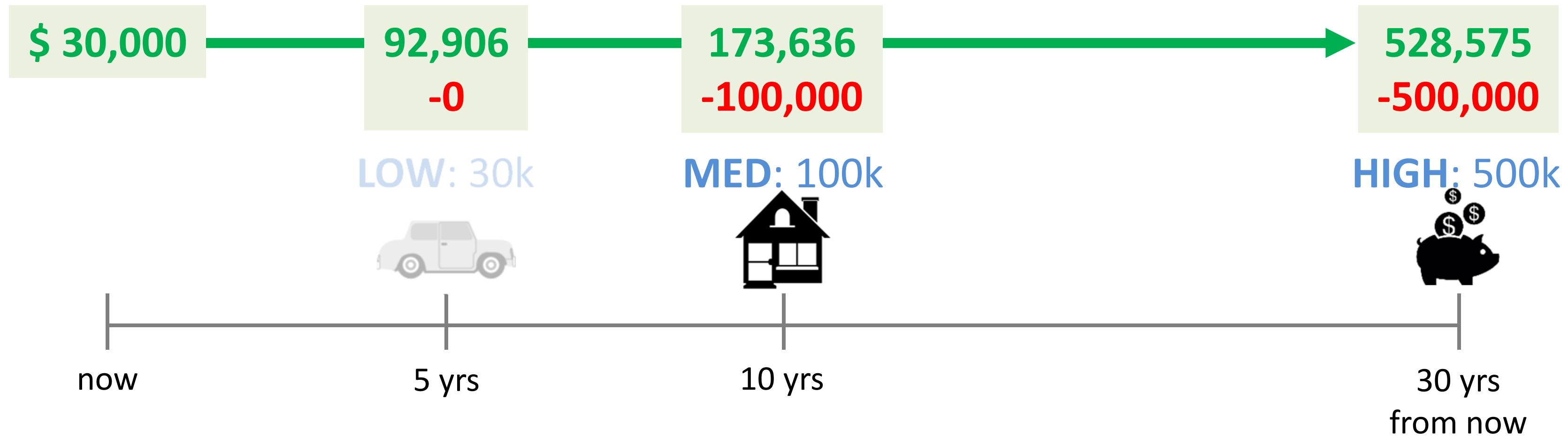


Advice: Save More

- It will be helpful if the individual has information on possible outcomes

- **Advice (2): Increase savings**
  - Increase monthly savings to $800
  - Skip purchasing a car

  ➔ Then, both high and medium goals can be achieved

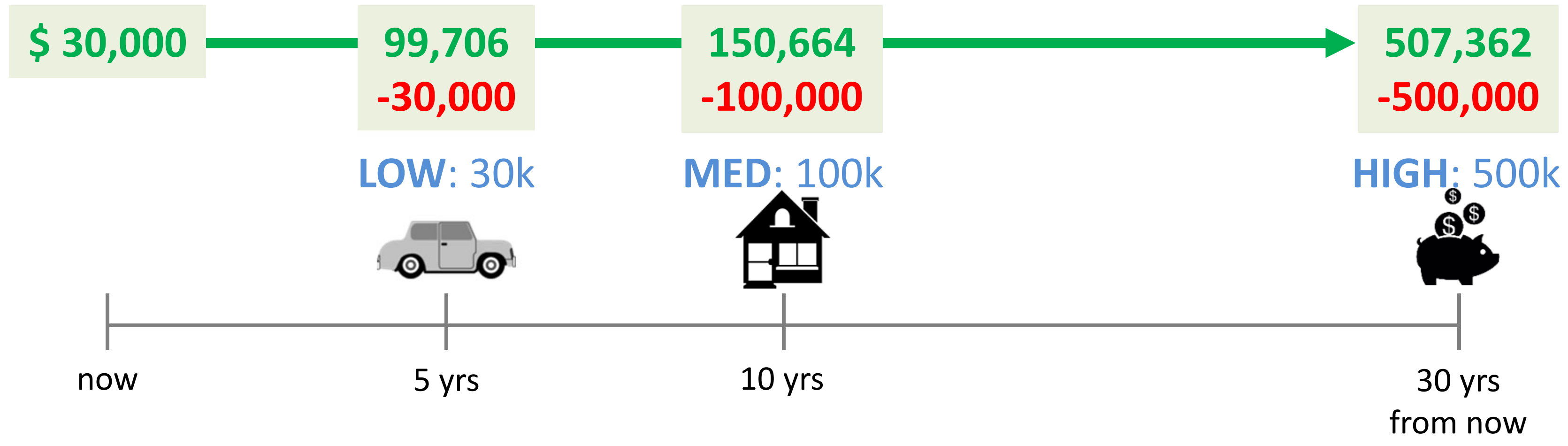


Advice: Save More

- It will be helpful if the individual has information on possible outcomes

- **Advice (3): Increase savings slightly more**
  - Increase monthly savings to $900

  ➔ **All goals can be achieved!**



Financial Planning Problem

- **Financial Planning:** (re)positioning of funds to achieve specific goals (Mulvey, 1992)

- Financial planning problems are ubiquitous
  - Sometimes called portfolio optimization, asset liability management (ALM), goal-based investing (GBI), liability-driven investment (LDI) depending on the context.

Subject	Financial planning goal
Commercial Banks	Maintain sufficient liquidity of assets while maximizing profit
Pension Funds	Manage asset to meet liabilities while reducing contribution cost.
Insurance Companies	Deal with randomly occurring insurance claims while maximizing profit
Individuals	Minimize cost to achieve certain goals at predetermined time.

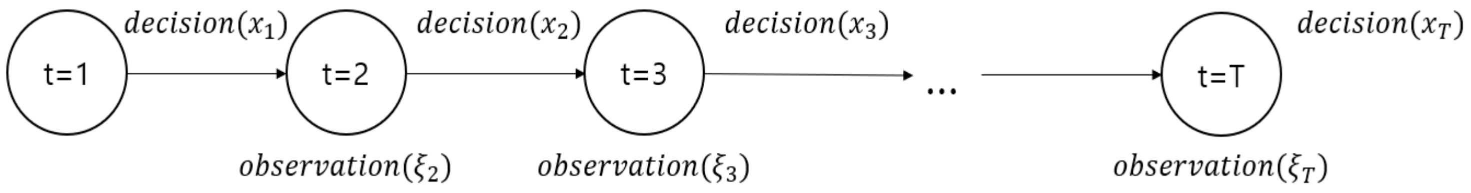
Prof. Dempster's Pioneering Works on Financial Planning

- Dynamic stochastic programming for asset-liability management. Annals of Operations Research 81, 131–162 (1998)
- Global Asset Liability Management. British Actuarial Journal, 9(1), 137-195 (2003)
- Sequential Importance Sampling Algorithms for Dynamic Stochastic Programming. J Math Sci 133, 1422–1444 (2006)
- Necessary and sufficient optimality conditions for control of piecewise deterministic Markov processes, Stochastics and Stochastic Reports, 40:3-4, 125-145 (1992)
- Large-Scale Linear Programming. IIASA Collaborative Paper. IIASA, Laxenburg, Austria: CP-81-S1 (1981)
- Dynamic portfolio replication using stochastic programming, in Risk management: Value at risk and beyond, 100-128 (2002)
- Asset liability management for individual households - Abstract of the London Discussion. British Actuarial Journal, 16(2), 441-467 (2011)
- Designing minimum guaranteed return funds, Quantitative Finance, 7:2, 245-256 (2007)
- Managing guarantees, The Journal of Portfolio Management, 51-61 (2006)
- Generalized Bellman-Hamilton-Jacobi optimality conditions for a control problem with a boundary condition. Appl Math Optim 33, 211–225 (1996)
- The CALM stochastic programming model for dynamic asset-liability management in "World Wide Asset and Liability Modelling", 464-500 (1998)
- And more…

Multistage Stochastic Optimization Problem

[Multistage Stochastic Optimization Problem]

Multi-period sequential decision making problems under uncertainty



⇩

Multistage Stochastic Optimization Problems (MSOP)

- Decision Process $x_{[T]} = (x_1, x_2, \dots, x_T)$.
- Stochastic Process $\xi_{[T]} = (\xi_1, \xi_2, \dots, \xi_T)$.

Multistage Stochastic Optimization Problem

[Multistage Stochastic Optimization Problem]

Nested form of MSOP

$$\min_{x1\in \mathcal{X}_1} f_1(x_1) + \mathbb{E}[\min_{x2\in \mathcal{X}_2(x_1,\xi_2)} f_2(x_2,\xi_2) + \mathbb{E}_{\cdot|\xi_{[2]}}[\cdots + \mathbb{E}_{\cdot|\xi_{[T-1]}}[\min_{xT\in \mathcal{X}_T(x_{T-1},\xi_T)} f_T(x_T,\xi_T)]]]$$

- $\mathbb{E}_{\cdot|\xi_{[t]}}$: Conditional expectation operator with respect to $\xi_{[t]}$.
- $f_t(x_t,\xi_t)$: Convex objective function in x_t and dependent on ξ_t.
- $\mathcal{X}_t(x_{t-1},\xi_t)$: Convex feasible region for x_t given x_{t-1} and ξ_t.

Multistage Stochastic Programming

[Multistage Stochastic Optimization Problem]

Nested form of MSOP

$$\min_{x1\in \mathcal{X}_1} f_1(x_1) + \mathbb{E}[\min_{x2\in \mathcal{X}_2(x_1,\xi_2)} f_2(x_2,\xi_2) + \mathbb{E}_{\cdot|\xi_{[2]}}[\cdots + \mathbb{E}_{\cdot|\xi_{[T-1]}}[\min_{xT\in \mathcal{X}_T(x_{T-1},\xi_T)} f_T(x_T,\xi_T)]]]$$

- It can be solved by Multistage Stochastic Programming (MSP).

- The usual MSP approach solves MSOP by
  1. Constructing a scenario tree that approximates the stochastic process $\xi_{[T]}$ with finite number of realizations
  2. Solving a large deterministic equivalent convex optimization problem under the realized scenario tree.

- Drawbacks of MSP: Curse of dimensionality
  - The number of scenarios increases exponentially with respect to the number of stages and/or the number of nodes per stage.
  - Problems often become intractable.

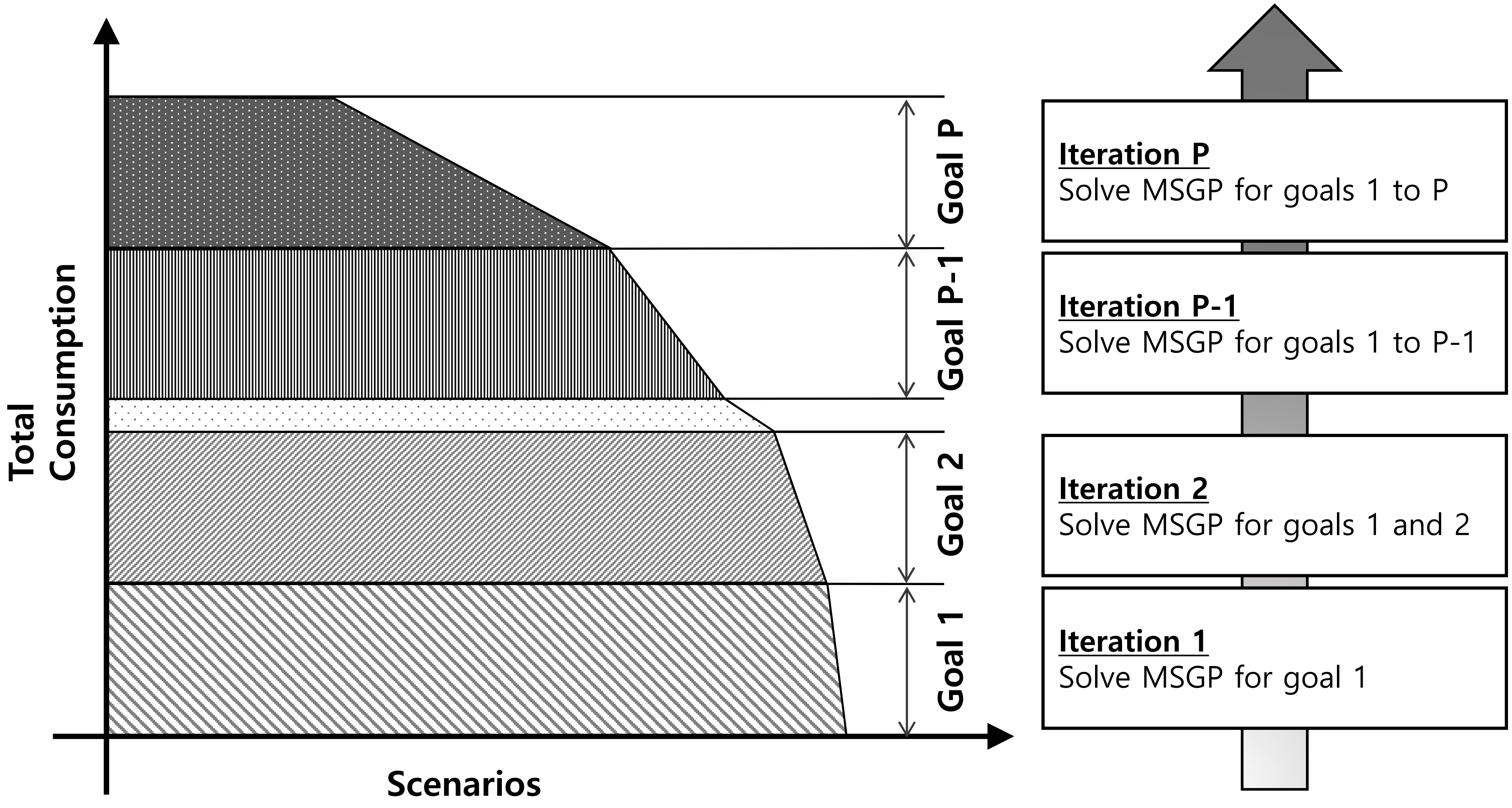
Multi-stage Stochastic Goal Programming (MSGP)

- **MSGP Procedure**
  - Problems are solved ***sequentially*** to preferentially satisfy consumption goals with ***higher priorities***

- Example)

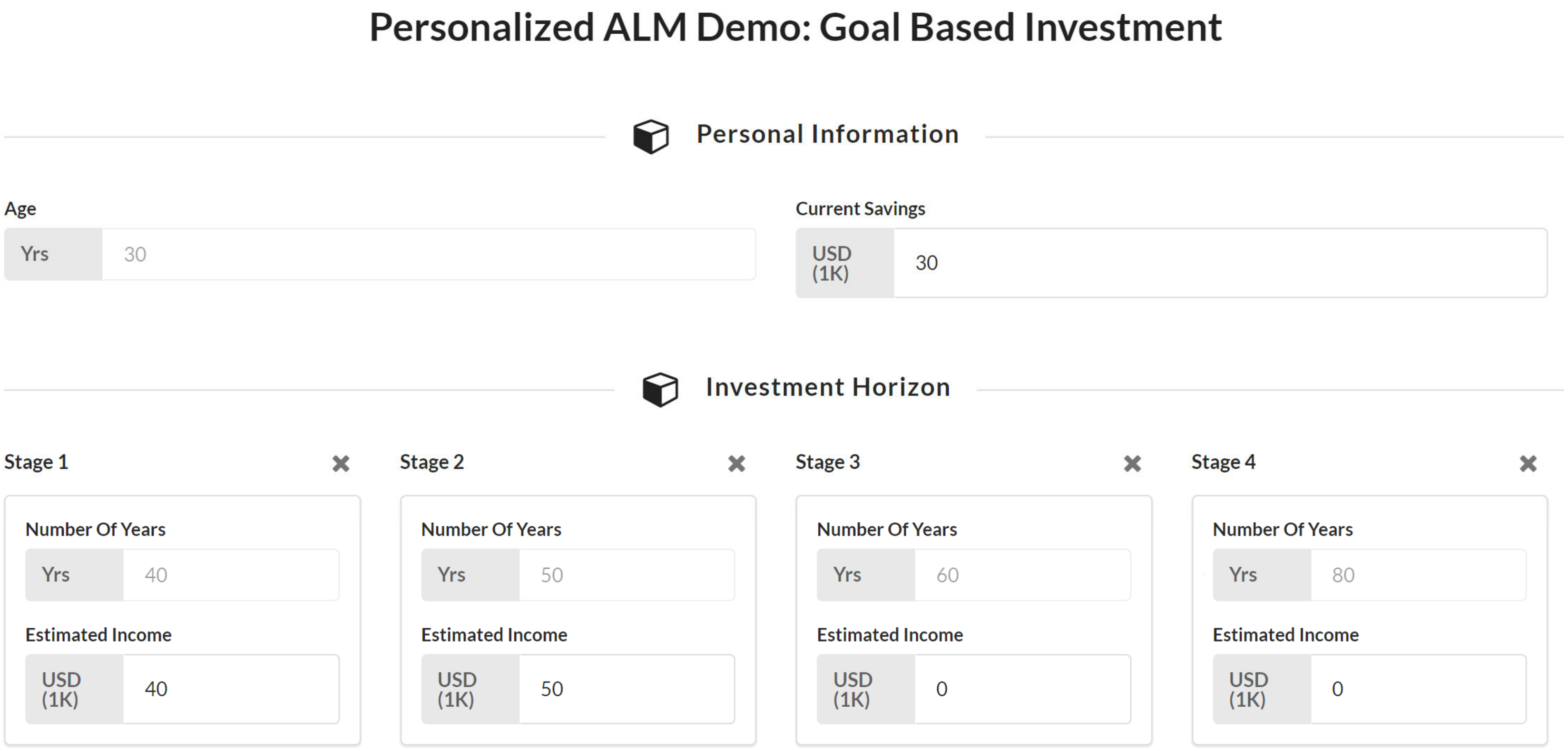
Goals	MSP Problems	Optimization Results
1) **Post-Retirement Living** (priority 1)	**Maximize** Prob (Goal 1) **Subject to** Constraints	***MaxProb (Goal 1)***
2) **Child Education** (priority 2)	**Maximize** Prob (Goal 2) **Subject to** Constraints ***MaxProb (Goal 1)***	***MaxProb (Goal 2)***
3) **Luxury Life** (priority 3)	**Maximize** Prob (Goal 3) **Subject to** Constraints ***MaxProb (Goal 1)*** ***MaxProb (Goal 2)***	***MaxProb (Goal 3)***
4) …		

MSGP for Goal Based Investment



Example

- Age, Savings, and Income



Personalized ALM Demo: Goal Based Investment

Personal Information

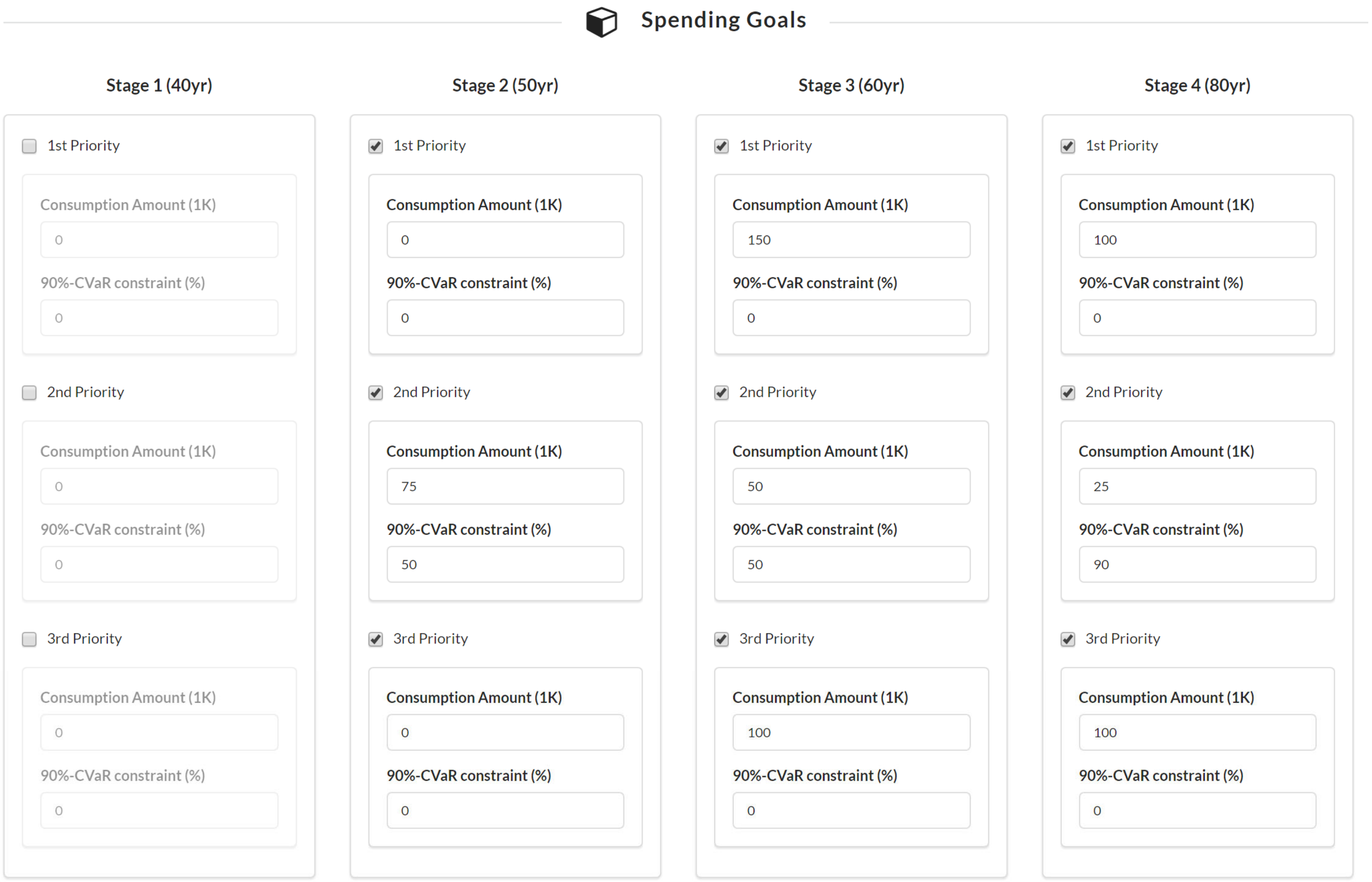
Age	Yrs 30
Current Savings	USD (1K) 30

Investment Horizon

	Stage 1	Stage 2	Stage 3	Stage 4
Number Of Years (Yrs)	40	50	60	80
Estimated Income (USD (1K))	40	50	0	0

Example

- Goals

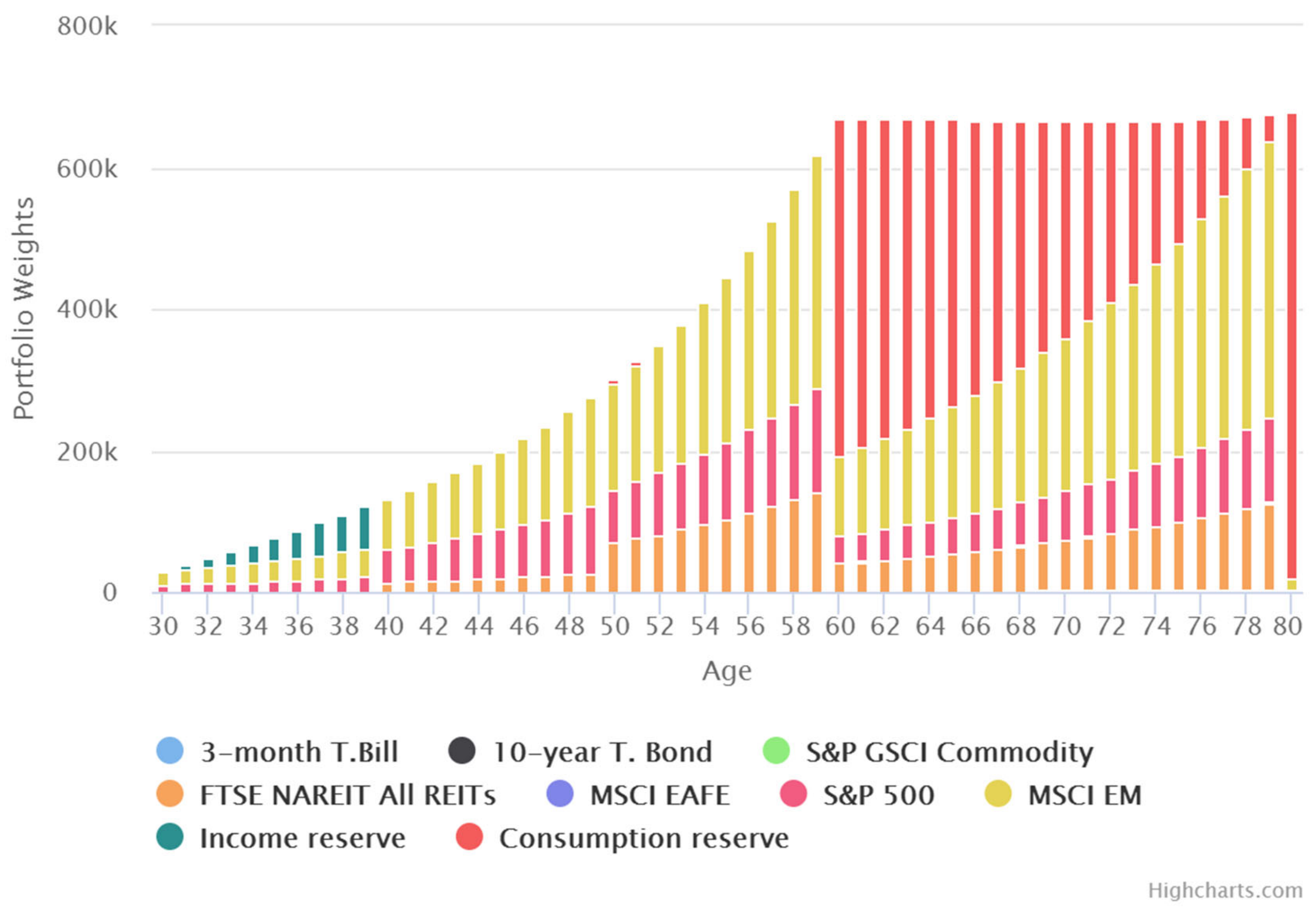


Spending Goals

	Stage 1 (40yr)	Stage 2 (50yr)	Stage 3 (60yr)	Stage 4 (80yr)
1st Priority	☐	✓	✓	✓
Consumption Amount (1K)	0	0	150	100
90%-CVaR constraint (%)	0	0	0	0
2nd Priority	☐	✓	✓	✓
Consumption Amount (1K)	0	75	50	25
90%-CVaR constraint (%)	0	50	50	90
3rd Priority	☐	✓	✓	✓
Consumption Amount (1K)	0	0	100	100
90%-CVaR constraint (%)	0	0	0	0

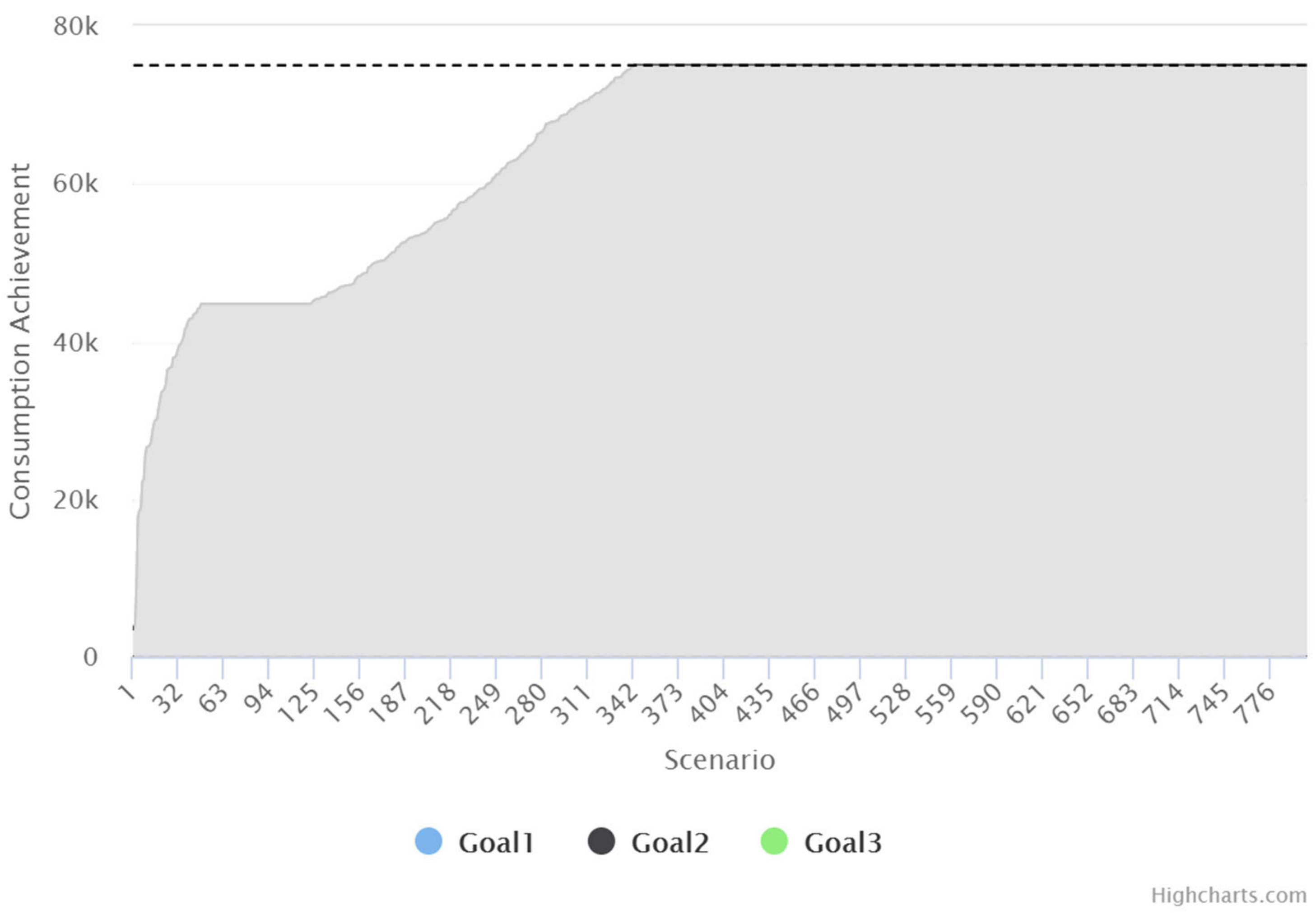
Example

- Expected Wealth with Asset Allocation



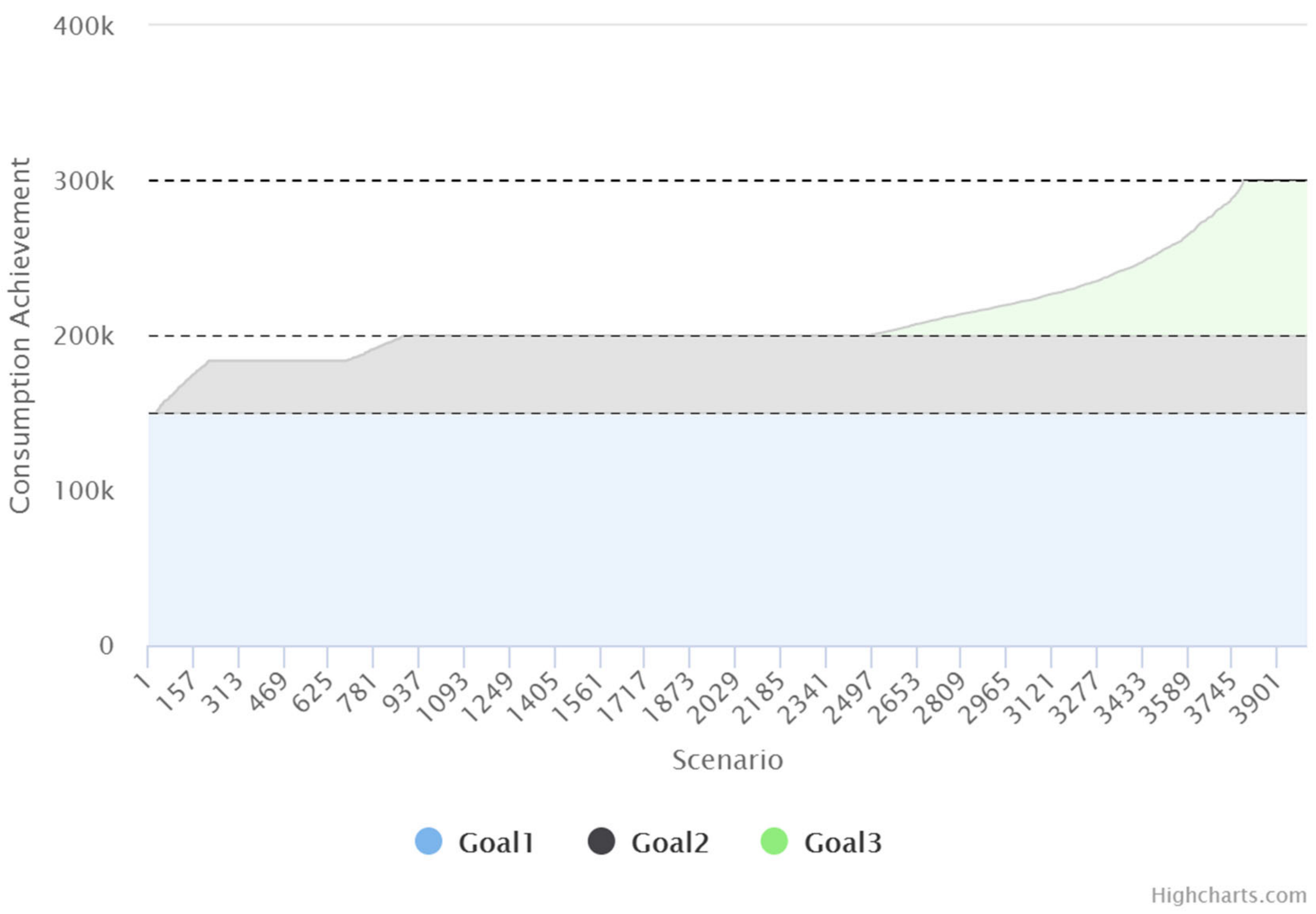
Example

- Goal Achievement – Goal 2 (Stage 2)



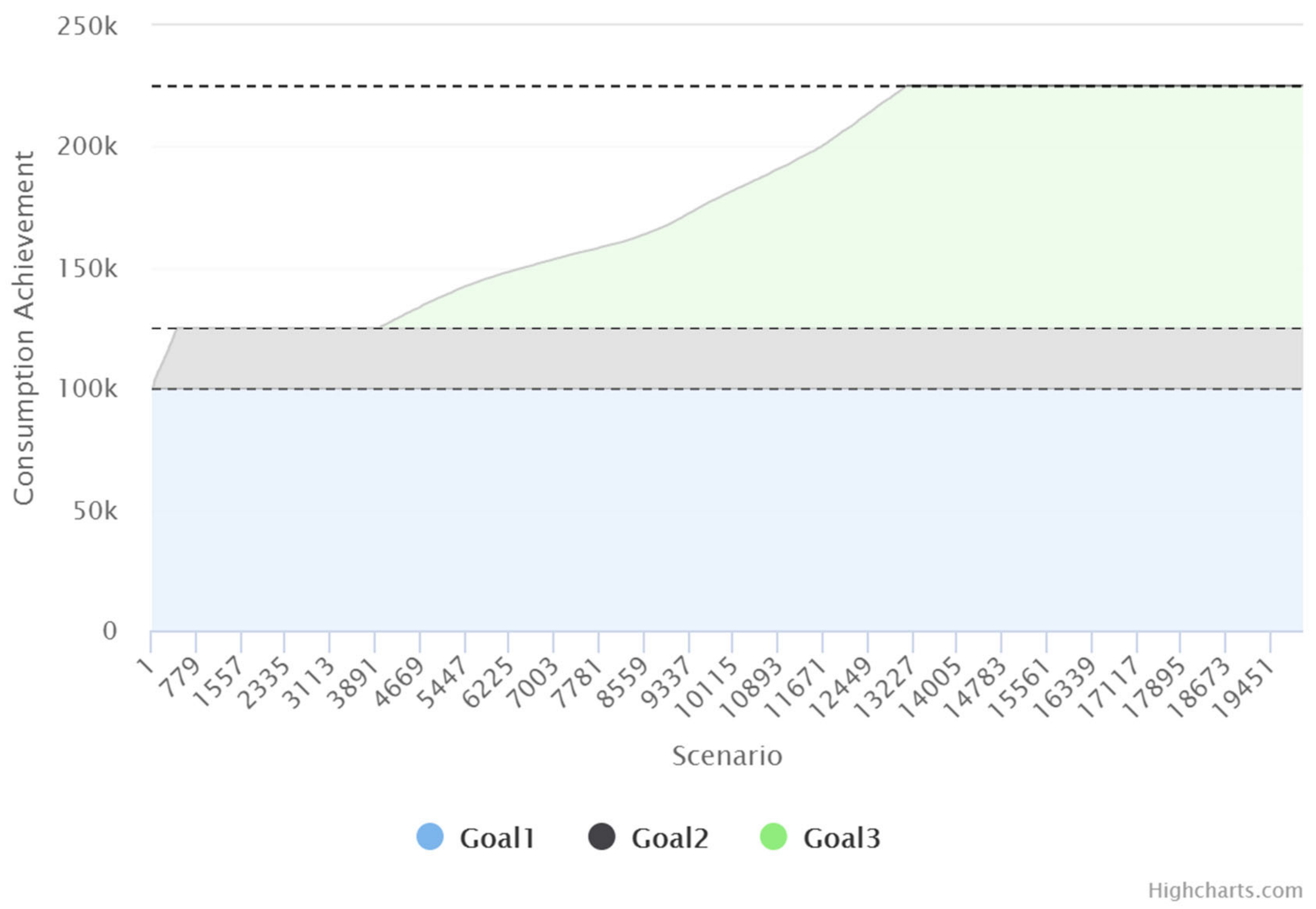
Example

- Goal Achievement – Goals 1, 2, & 3 (Stage 3)



Example

- Goal Achievement – Goals 1, 2 & 3 (Stage 4)



That’s Nice, but…

- More stages are needed
- Computational time should be further reduced

Part 2

APPLYING MACHINE LEARNING TECHNIQUES TO LARGE-SCALE MULTI-STAGE STOCHASTIC PROGRAMMING ALGORITHM

Recall: Multistage Stochastic Programming

[Multistage Stochastic Optimization Problem]

Nested form of MSOP

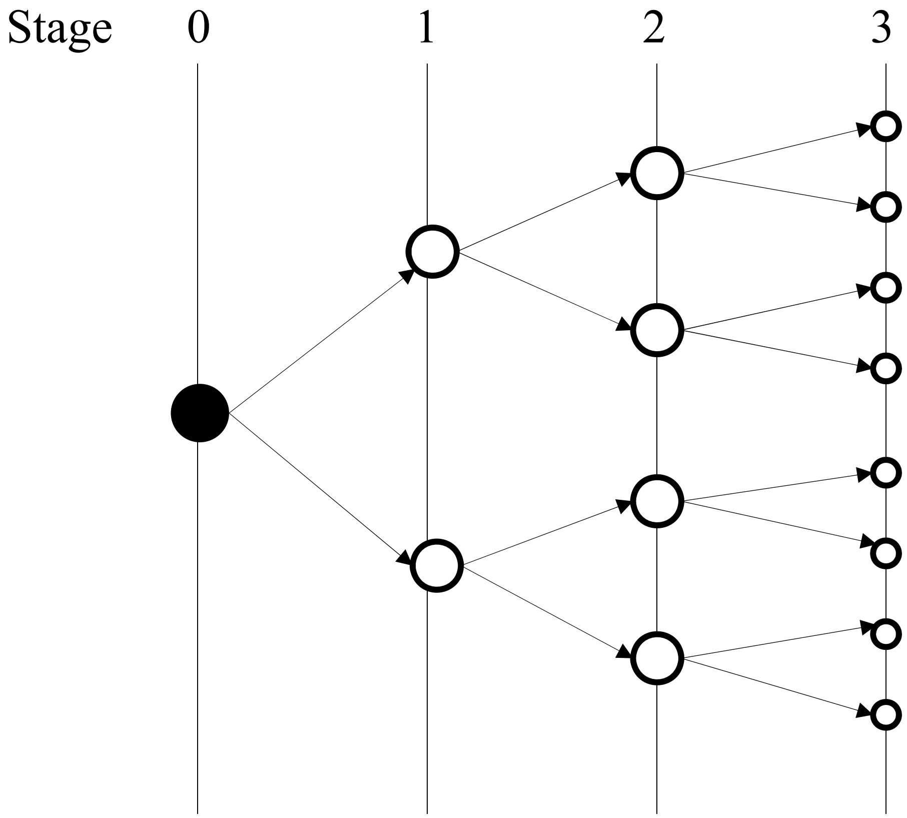
$$\min_{x1 \in \mathcal{X}_1} f_1(x_1) + \mathbb{E}\left[\min_{x2 \in \mathcal{X}_2(x_1,\xi_2)} f_2(x_2,\xi_2) + \mathbb{E}_{\cdot|\xi_{[2]}}\left[\cdots + \mathbb{E}_{\cdot|\xi_{[T-1]}}\left[\min_{xT \in \mathcal{X}_T(x_{T-1},\xi_T)} f_T(x_T,\xi_T)\right]\right]\right]$$

- It can be solved by Multistage Stochastic Programming (MSP).
- The usual MSP approach solves MSOP by
  1. Constructing a scenario tree that approximates the stochastic process $\xi_{[T]}$ with finite number of realizations
  2. Solving a large deterministic equivalent convex optimization problem under the realized scenario tree.

- Drawbacks of MSP: Curse of dimensionality
  - The number of scenarios increases exponentially with respect to the number of stages and/or the number of nodes per stage.
  - Problems often become intractable.

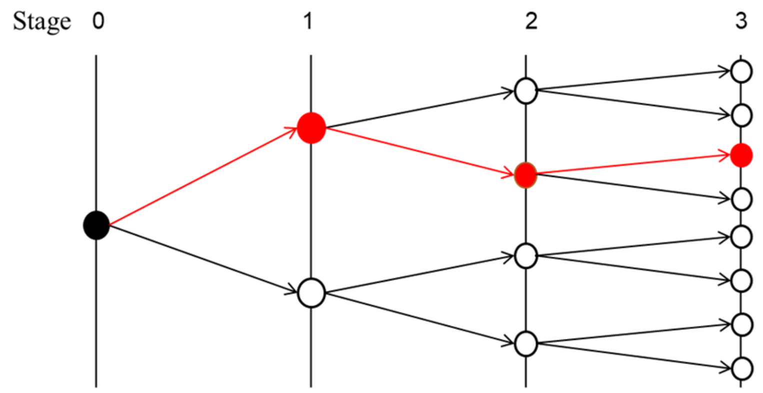
Curse-of-dimensional Issue of MSP

- MSP algorithm solves a huge deterministic equivalent convex problem
- Size of scenario tree grows exponentially to the number of stages
- Reducing the number of stages can make the problem unrealistic

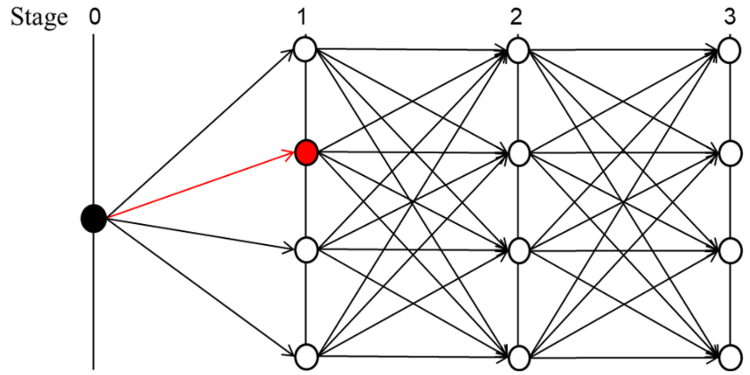


Decomposition Methods of Stochastic Programs

- **Scenario Decomposition**
  - Decompose problems scenario-wise
  - Non-anticipativity constraint relaxed
  - <u>Requires loops over the entire scenario!</u>
    - Not suitable for problems with extremely large scenario tree
  - Ex) dual decomposition, progressive hedging



- **Stagewise Decomposition**
  - Decompose the problem stagewise
  - Consequence of current decision is summarized by value function.
  - Goal is to approximate the value function accurately and efficiently
  - Often requires special structure in the problem for the efficiency
  - Ex) nested Benders decomposition, SDDP



Stagewise Decomposition

- Stagewise Decomposition Algorithm
  - Decompose the MSOP problem into stagewise subproblems.

[Stagewise Subproblems]

For $t = 1$,

$$\min_{x1\in \mathcal{X}_1} f_1(x_1) + Q_2(x_1)$$

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

$$\min_{x_t\in\mathcal{X}_t(x_{t-1},\xi_t)} f_t(x_t, \xi_t) + Q_{t+1}\left(x_t, \xi_{[t]}\right)$$

- where the value function Q_t is recursively defined by the Bellman Equation.
  - For $t = T, \dots, 2$,
  - $\mathcal{Q}_t(x_{t-1}, \xi_{[t]}) = \min_{x_t\in\mathcal{X}_t(x_{t-1},\xi_t)} f_t(x_t, \xi_t) + Q_{t+1}\left(x_t, \xi_{[t]}\right)$
  - $Q_{t+1}\left(x_t, \xi_{[t]}\right) \coloneqq \mathbb{E}_{\cdot|\xi_{[t]}}[\mathcal{Q}_{t+1}\left(x_t, \xi_{[t+1]}\right)]$
  - $Q_{T+1} \equiv 0$

Stagewise Decomposition

- Stagewise Decomposition Algorithm
  - Decompose the MSOP problem into stagewise subproblems.

[Stagewise Subproblems]

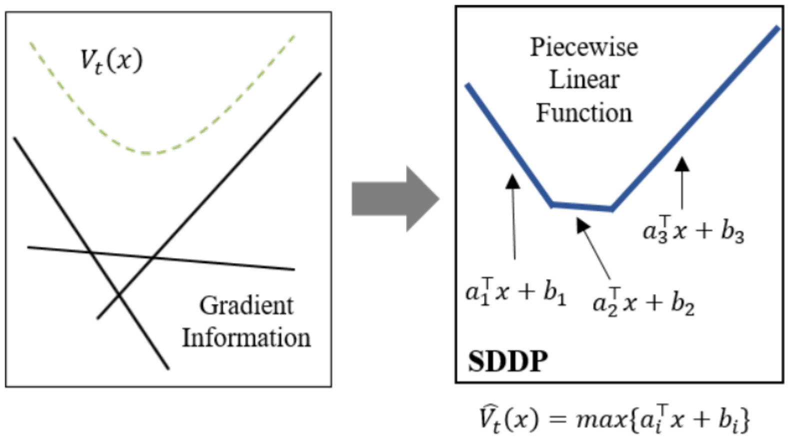
$$\min_{x1\in \mathcal{X}_1} f_1(x_1) + \mathbb{E}\underbrace{[\min_{x2\in \mathcal{X}_2(x_1,\xi_2)} f_2(x_2,\xi_2) + \underbrace{\mathbb{E}_{\cdot|\xi_{[2]}}[\cdots + \underbrace{\mathbb{E}_{\cdot|\xi_{[T-1]}}[\min_{xT\in \mathcal{X}_T(x_{T-1},\xi_T)} f_T(x_T,\xi_T)]]]}_{Q_T(x_{T-1},\,\xi_{[T-1]})}}_{Q_3(x_2,\,\xi_{[2]})}}_{Q_2(x_1)}$$

Stochastic Dual Dynamic Programming

- Stochastic Dual Dynamic Programming (SDDP)
  - State-of-the-art stagewise decomposition algorithm for solving large-scale multistage stochastic optimization problems
  - SDDP was introduced by Pereira & Pinto (1991)

- SDDP solves stagewise subproblems by approximating the value function as a piecewise linear convex function based on Benders decomposition.

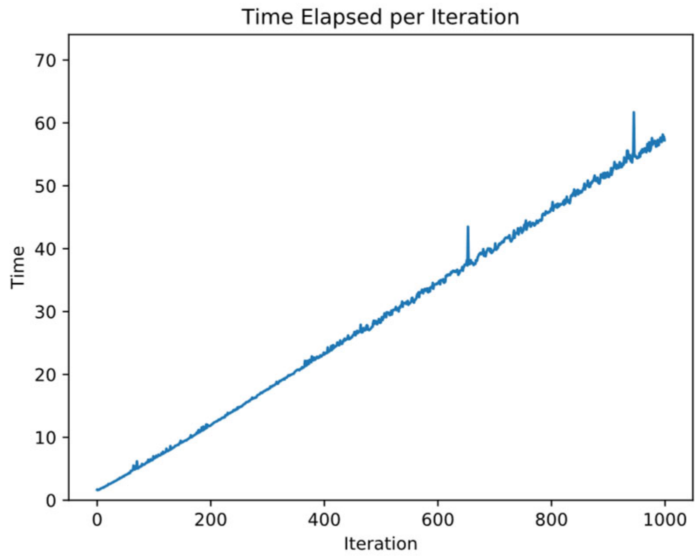
- The piecewise linear convex function is improved gradually by using subgradient cutting planes (called cuts) as iterating the algorithm.



SDDP: Issues

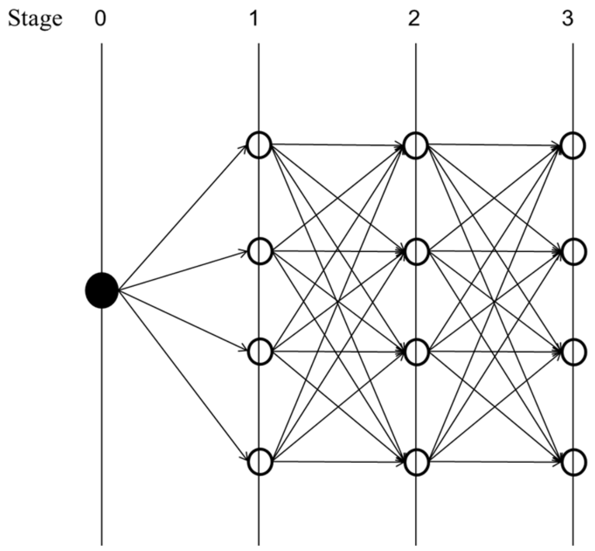
Increasing Computational Burden

- Optimality cut is added every iteration
- Size of subproblem increases every iteration
- Computation time of subproblem increases every iteration



Distributional Approximation with Scenario Tree

- Distribution of underlying stochastic process has to be approximated with scenario tree



Improving SDDP: Selection of Cuts

- Main idea: Constructing a piecewise linear lower bound of the value function using selected cuts that **satisfy heuristic conditions.**
  - Reduce the size of the subproblems.

- Pfeiffer et al. (2012) proposed the **territory algorithm** and **test of usefulness.**
  - Territory algorithm selects candidates of potentially useless cuts.
  - And redundant cuts are removed after verifying usefulness within candidate group by test of usefulness.
  - Utilizing two algorithms is difficult to use in practice because of high computational cost.

- De Matos el al. (2015) proposed the **Level N dominance** cut selection strategy.
  - It constructs a piecewise linear lower bound by selecting cuts activated at least n for the trial solutions.
  - They concluded that the Level 1 dominance outperforms.

Improving SDDP: Generation of Cuts

- Dai et al. (2021) proposed **$\boldsymbol{\nu}$-SDDP.**

- In ν-SDDP, neural network is trained using a meta-learning approach to **generate a fixed number of corresponding cuts** based on the problem context vector.

- This is the novel algorithm that generates cuts using neural network.

- Drawbacks:
  – Previously generated cuts are not considered to generate new ones.
  – Only linear programs can be solved by ν-SDDP.
  – The number of cuts to be generated is fixed.

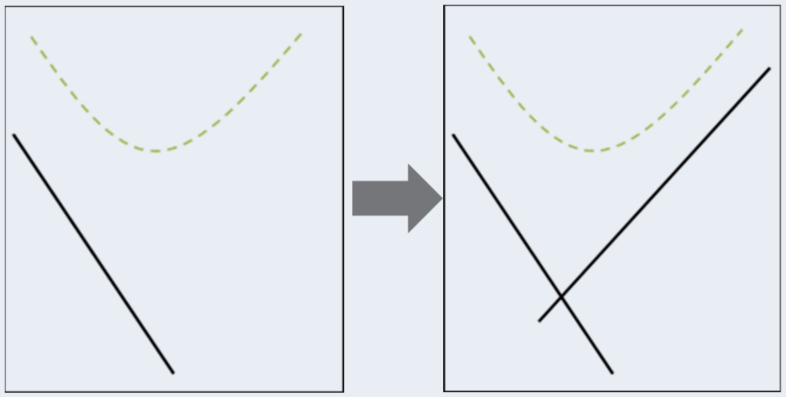
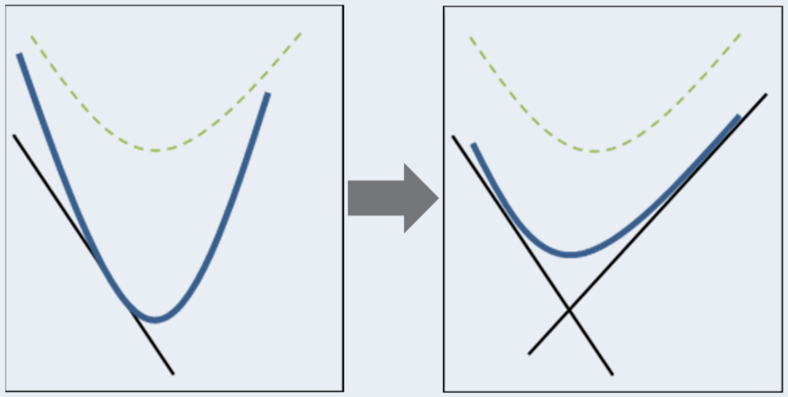
Part 2-1

PARAMETRIC VALUE FUNCTION APPROXIMATION FOR LARGE-SCALE MULTISTAGE STOCHASTIC PROGRAMMING PROBLEMS

Based on:
Jinkyu Lee, Sanghyeon Bae, Woo Chang Kim, Yongjae Lee, 2023, Value function gradient learning for large-scale multistage stochastic programming problems, *European Journal of Operational Research*, 308.1, 321-335

Value Function Gradient Learning (VFGL)

- **Key Idea:** Fix the parametric families of value function and learn the parameter that well approximates the gradient of value function
- **Key Characteristics:**
    - No extra constraint (optimality cut) is added to the problem.
    - Samples directly from the underlying distribution

Piecewise-linear approximation on the value function	VFGL
$\widehat{V}_t(x) = \max\{a_i^{\mathrm{T}} x + b_i\}_i$	$\widehat{V}_t(x) = g_t(x\|\theta_t)$
	

VFGL Choice of the Parametric Families

- **Observation 1**: For each subproblem, a previous decision variable is often primarily involved in a *resource constraint*
  - Ex) Asset realization constraint, water reservoir level, storage level
  - $R_t = k(x_{t-1}, \xi_t)$

- **Observation 2**: For each subproblem, the objective is often (directly/indirectly) a function of available resource at that time
  - $stage\ t\ objective = f_t\big(k(x_{t-1}, \xi_t)\big)$

Claim: A good heuristic starting point for the value function parametric families is an indefinite integral form of the stagewise objective function

VFGL KKT Condition: with the Approximated VF

Stationarity:

$$\nabla f_t(x_t) + \nabla \hat{V}_{t+1}(x_t|\theta_{t+1}) - \sum_{i=1}^{k} \lambda_{t,i} \nabla k_{t,i}(x_t) + \sum_{i=1}^{p} \mu_{t,i} \nabla h_{t,j}(x_t) = 0$$

Primal Feasibility:

$k_{t,i}(x_t) \le b_{t,i}(x_{t-1}, \xi_t)$ $\quad i = 1, \dots, k_{\mathrm{t}}$

$h_{t,j}(x_t) = -d_{t,j}(x_{t-1}, \xi_t)$ $\quad j = 1, \dots, p_{\mathrm{t}}$

Dual Feasibility:

$\mu_{t,i} \ge 0, i = 1, \dots, k$

Complementary Slackness:

$\mu_{t,i}\left(k_{t,i}(x_t) - b_{t,i}(x_{t-1}, \xi_t)\right) = 0, i = 1, \dots, k_{\mathrm{t}}$

VFGL KKT Condition: with the Approximated VF

Stationarity:

$$\nabla f_t(x_t) + \nabla V_{t+1}(x_t) + \sum_{i=1}^{k} \hat{\lambda}_{t,i} \nabla g_{t,i}(x_t) + \sum_{i=1}^{p} \hat{\mu}_{t,i} \nabla h_{t,j}(x_t) = \nabla V_{t+1}(x_t) - \nabla \hat{V}_{t+1}(x_t|\theta_{t+1})$$

$$\downarrow$$

Closer the $\nabla V_{t+1}(x_t) - \nabla \hat{V}_{t+1}(x_t|\theta_{t+1})$ is to 0, better the approximation $\hat{V}_{t+1}(x_t|\theta_{t+1})$

Primal Feasibility:

$$k_{t,i}(x_t) \le b_{t,i}(x_{t-1}, \xi_t) \qquad i = 1, \dots, k_\text{t}$$

$$h_{t,j}(x_t) = -d_{t,j}(x_{t-1}, \xi_t) \qquad j = 1, \dots, p_\text{t}$$

Dual Feasibility:

$$\mu_{t,i} \ge 0, i = 1, \dots, k$$

Complementary Slackness:

$$\mu_{t,i}\left(k_{t,i}(x_t) - b_{t,i}(x_{t-1}, \xi_t)\right) = 0, i = 1, \dots, k_\text{t}$$

VFGL Loss Function

Define loss function to minimize $J_{t+1}(\theta_{t+1}; x_t) = \left\| \nabla V_{t+1}(x_t) - \nabla \hat{V}_{t+1}(x_t | \theta_{t+1}) \right\|$

$$\mathrm{J}_{t+1}(\theta_{t+1}) = \mathbb{E}_{x_t}[J_{t+1}(\theta_{t+1}; x_t)]$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} J_{t+1}(\theta_{t+1}, x_t^i)$$

Optimize $\mathrm{J}_{t+1}(\theta_{t+1})$ by the stochastic gradient descent method

Stochastic Gradient Descent

1. Initialize $i \leftarrow 1$ (iteration counter), α_i (step size)
2. Sample $\nabla_{\theta_{t+1}} J_{t+1}(\theta_{t+1}, x_t^i) \approx \frac{1}{K} \sum_{s=1}^{K} \nabla_\theta \mathcal{J}_{t+1}(\theta_{t+1}; x_t^i, \xi_{t+1}^s)$
3. Update $\theta_{t+1} \leftarrow \theta_{t+1} - \alpha_i \nabla_{\theta_{t+1}} J_{t+1}(\theta_{t+1}, x_t^i)$
4. $i \leftarrow i + 1$
5. Go to step 2 until θ_{t+1} is sufficiently converged

where α_i satisfies $\sum_{i=1}^{\infty} \alpha_i = \infty, \sum_{i=1}^{\infty} \alpha_i^2 = \infty$

VFGL Sampling Target Gradient

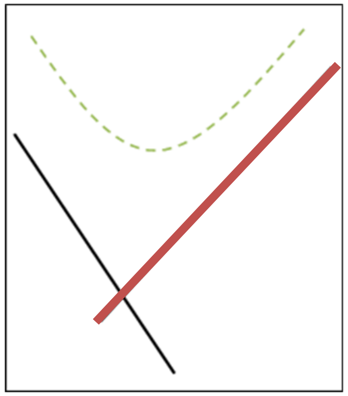
- But how do we sample the target gradient $\frac{\partial}{\partial x_{t,n}} V_{t+1}(x_t, \xi_{t+1})$?
- **Differentiate the Lagrangian** at optimal

Suppose that we have primal dual optimal solution $(x_t^*, \lambda_t^*, \mu_t^*)$ with $(\tilde{x}_{t-1}, \tilde{\xi}_t)$

$$\begin{aligned} \nabla_{x_{t-1}} \hat{V}_t(x_{t-1}, \xi_t) &= \mathrm{L}\left(\tilde{x}_{t-1}, \tilde{\xi}_t | x_t^*, \lambda_t^*, \mu_t^*\right) \\ &= \sum_{i=1}^{k} \lambda_{t,i}^* \nabla_{x_{t-1}} b_{t,i}(x_{t-1}, \xi_t) + \sum_{i=1}^{p} \mu_{t,i}^* \nabla_{x_{t-1}} d_{t,j}(x_{t-1}, \xi_t) \end{aligned}$$

- But $V_{t+1}(x_t^*)$ is required to find $(x_t^*, \lambda_t^*, \mu_t^*)$
  - **Bootstrap** V_{t+1} with $\hat{V}_{t+1}$

$\curvearrowright$ Gradient of the SDDP optimality cut



VFGL Objective Weighting

- Our loss function does not consider the scale of value function gradients.
  - Can cause bias towards decision variable with huge gradient scales

- Define the **average value function gradient**:

$$v_{t+1,n} = E_{\xi_{[t+1]}}\left[\frac{\partial}{\partial x_{t,n}} V_{t+1}(x_t, \xi_{t+1})\right]$$

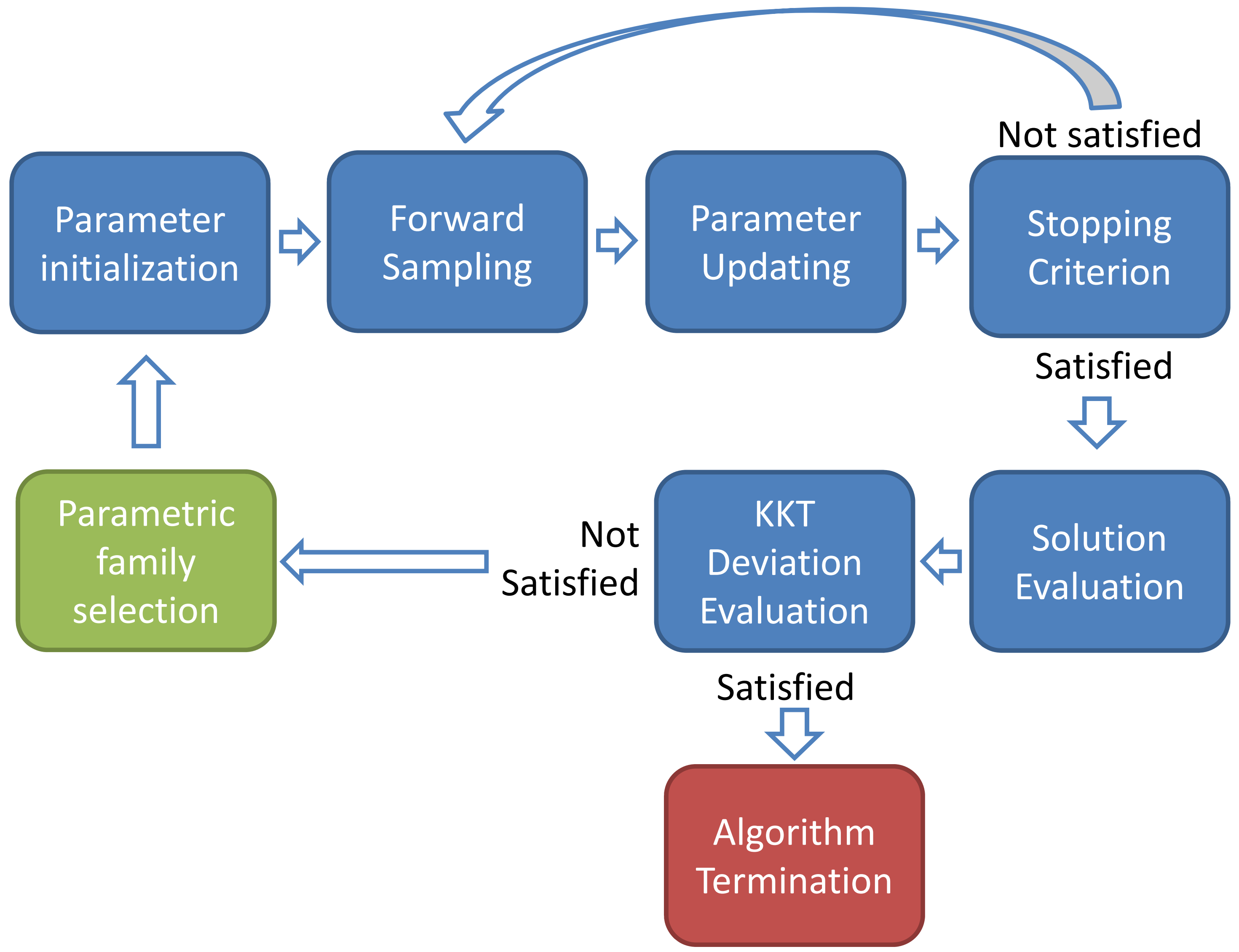
- **Scale the loss function** by the average value function gradient

$$\mathrm{J}^{w}_{t+1}(\theta_{t+1}, x_t) = \sum_{n=1}^{n_t}\left(\frac{\frac{\partial}{\partial x_{t,n}} V_{t+1}(x_t) - \frac{\partial}{\partial x_{t,n}} \hat{V}_{t+1}(x_t|\theta_{t+1})}{v_{t+1,n}}\right)^2$$

- Since $\boldsymbol{v_{t+1,m}}$ is unobservable, we **approximate** this value by the sample mean and update the approximation

$$\bar{v}_{t+1,n} \leftarrow \frac{s}{s+1}\bar{v}_{t+1,n} + \frac{1}{s+1}\left(\frac{\partial}{\partial x_{t,n}} \hat{V}_{t+1}(x_t, \xi_{t+1})\right)$$

VFGL Framework Summary



Production Optimization

Objective: Optimize production to minimize the average total cost while all demands are met.

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

Parameter	
$xc_{t,i}$	Resource cost of product i production at stage t
$yc_{t,i}$	Cost of product i outsourcing at stage t
$sc_{t,i}$	Cost of product i storing from stage t to stage $t+1$
r_t	Maximum resource available at stage t

Random Variable	
$d_{t,i}$	Demand of product i at stage t

Production Optimization

Stage 1		
minimize	$\sum_i y_{1,i} yc_{1,i} + \sum_i s_{1,i} sc_{1,i}$	
subject to	$\sum_i x_{1,i} xc_{1,i} \leq r_1$	Resource limit
	$s_{1,i} = x_{1,i} + y_{1,i}$	Storage balance
	$x_{1,i}, y_{1,i}, s_{1,i} \geq 0$	Non-negativity

Stage t		
minimize	$\sum_i y_{t,i} yc_{t,i} + \sum_i s_{t,i} sc_{t,i}$	
subject to	$\sum_i x_{t,i} xc_{t,i} \leq r_t$	Resource limit
	$s_{t,i} = s_{t-1,i} + x_{t,i} + y_{t,i} - d_{t,i}$	Storage balance
	$x_{t,i}, y_{t,i}, s_{t,i} \geq 0$	Non-negativity

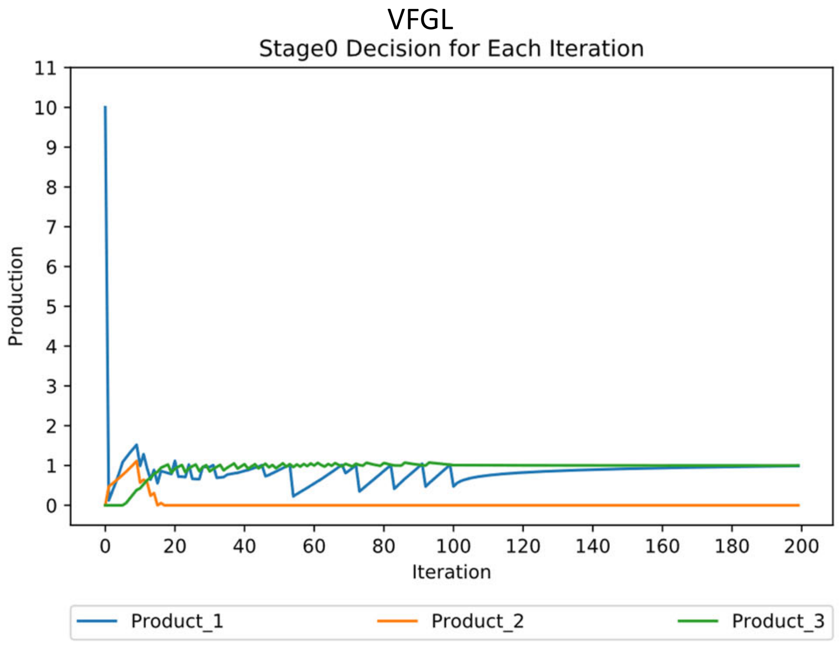
Parametric Functional Form

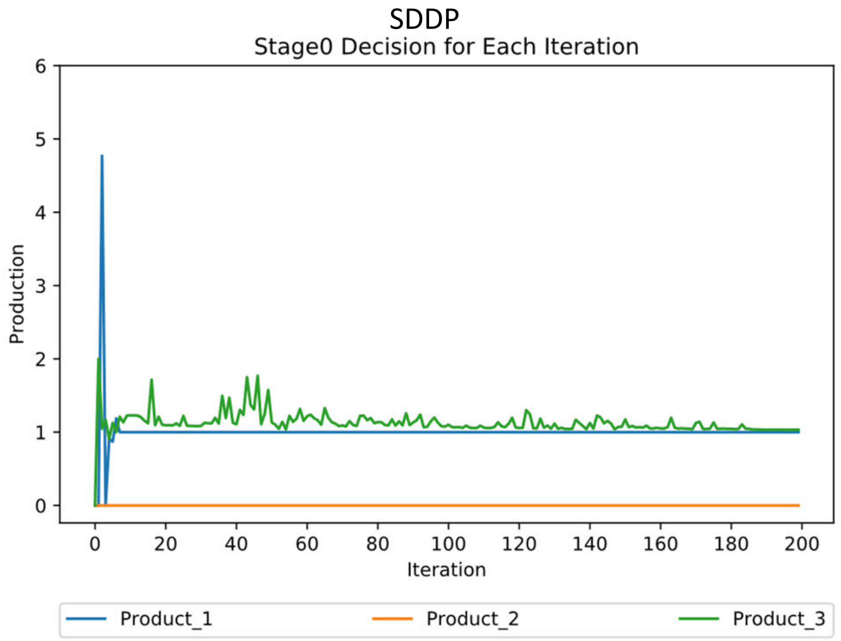
$$g_t\left(s_t | \theta_t = (\theta_{1,t}, \theta_{2,t}, \theta_{3,t})\right) = s_t \cdot \theta_{1,t} + \sum_{i=1}^{3} e^{-\theta_{2,t}[i] s_{t,i}}$$

Initial value of $\theta_t = (-1, -1, -1, 0, 0, 0)$

Production Optimization: Results

	Production			Outsource			Computation Time (s)
	Product 1	Product 2	Product 3	Product 1	Product 2	Product 3	
MSP	1.00	0.00	1.00	0.00	0.00	0.00	-
SDDP	1.00	0.00	1.06	0.00	0.00	0.00	1859
VFGL	0.99	0.00	1.00	0.00	0.00	0.00	227





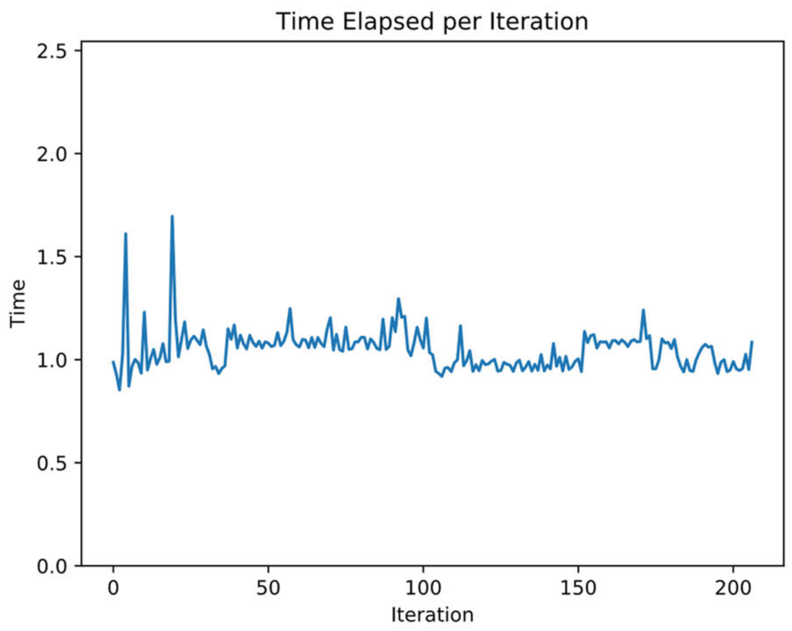
Production Optimization: Results

Parametric Function Form

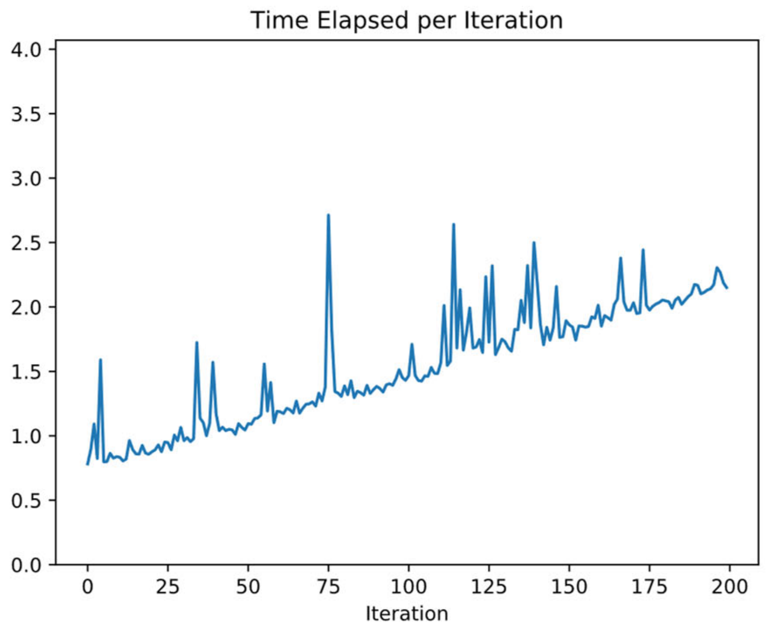
$$g_t\left(s_t|\theta_t = (\theta_{1,t}, \theta_{2,t})\right) = s_t \cdot \theta_{1,t} + \sum_{i=1}^{3} e^{-\theta_{2,t}[i] s_{t,i}}$$

Trained value of $\theta_0 = [-0.6559, -5.1594, -9.7626, 1.8402, -0.5000, -0.4367]$

VFGL



SDDP



Production Optimization: Perturbation

Maximum resource	**VFGL**					**MSP**			
	KKT deviation	Objective	Product 1	Product 2	Product 3	Objective	Product 1	Product 2	Product 3
8	1.22	227.63	1.63	0.33	1.07	227.22	2.00	0.00	1.20
8.5	1.85	215.83	1.61	0.09	1.07	214.65	2.50	0.00	1.10
9	0.98	202.38	1.58	0.00	1.02	202.44	2.00	0.00	1.00
9.5	1.35	190.08	1.49	0.00	1.01	190.39	1.50	0.00	1.00
*10	2.50	177.51	0.99	0.00	1.01	178.33	1.00	0.00	1.00
10.5	2.75	167.59	0.49	0.00	1.01	167.50	0.50	0.00	1.00
11	2.55	156.63	0.00	0.00	1.00	156.67	0.00	0.00	1.00
11.5	2.24	147.74	0.00	0.00	0.90	147.67	0.00	0.00	0.90
12	1.78	138.50	0.00	0.00	0.81	138.67	0.00	0.00	0.80
12.5	2.00	130.38	0.00	0.00	0.71	129.67	0.00	0.00	0.70
13	2.03	121.94	0.00	0.00	0.61	120.67	0.00	0.00	0.60
13.5	2.17	111.65	0.00	0.00	0.51	111.67	0.00	0.00	0.50
14	4.16	102.90	0.00	0.00	0.40	102.67	0.00	0.00	0.40
14.5	2.76	94.51	0.00	0.00	0.20	94.00	0.00	0.00	0.20
15	4.70	85.08	0.00	0.00	0.00	85.33	0.00	0.00	0.00

Hydro-thermal Energy Planning

Objective: Optimize energy production that minimizes cost while maximizing water level utility

Decision Variables	
r_t^{init}	Water reservoir level in the beginning of stage t
r_t^{final}	Water reservoir level in the end of stage t
H_t	Hydro electricity generation level at stage t
T_t	Thermal electricity generation level at stage t
Parameter	
r_0^{init}	Initial water reservoir level
c_t^H	Cost of hydro electricity production per unit at stage t
c_t^T	Cost of thermal electricity production per unit at stage t
d_t	Electricity demand at stage t
a_t	Reservoir level utility coefficient
b_t	Reservoir level utility scaling constant
Random Variable	
I_t	Water inflow to reservoir in the beginning of stage t

Hydro-thermal Energy Planning

Stage 1		
minimize	$c_1^H H_1 + c_1^T T_1 + e^{-a_1 r_1^{final} + b_1}$	
subject to	$r_1^{init} = r_0$	Initial reservoir
	$r_1^{final} = r_1^{init} - H_1$	Reservoir balance
	$H_1 + T_1 \geq d_1$	Demand
	$r_1^{final}, H_1, T_1 \geq 0$	Non-negativity

Stage t		
minimize	$c_t^H H_t + c_t^T T_t + e^{-a_t r_t^{final} + b_t}$	
subject to	$r_t^{init} = r_{t-1}^{final} + I_t$	Initial reservoir
	$r_t^{final} = r_t^{init} - H_t$	Reservoir balance
	$H_t + T_t \geq d_t$	Demand
	$r_t^{final}, H_t, T_t \geq 0$	Non-negativity

Parametric Function Form

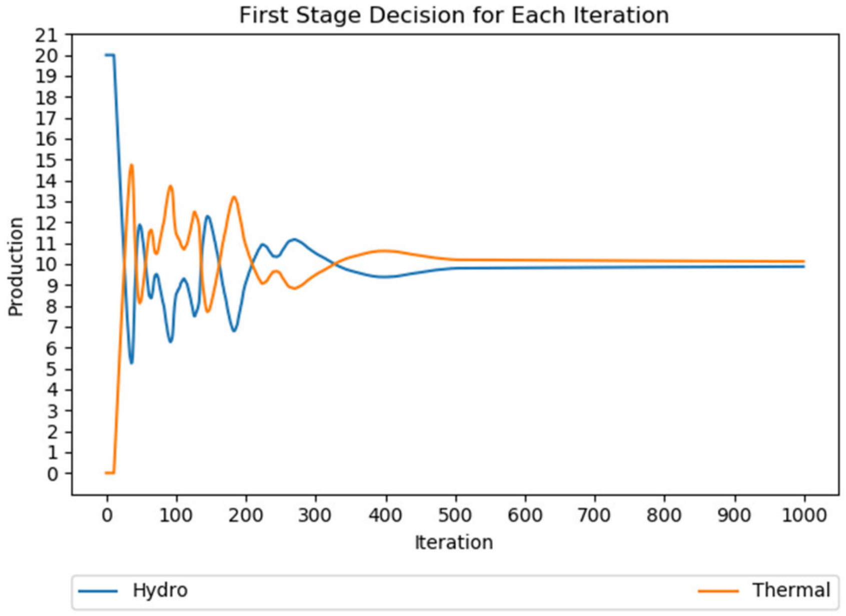
$$g_t\left(r_t | \theta_t = (\theta_{1,t}, \theta_{2,t})\right) = r_t \cdot \theta_{1,t} + e^{-\theta_{2,t} r_t + 5}$$

Initial value of $\theta_t = (1,1)$

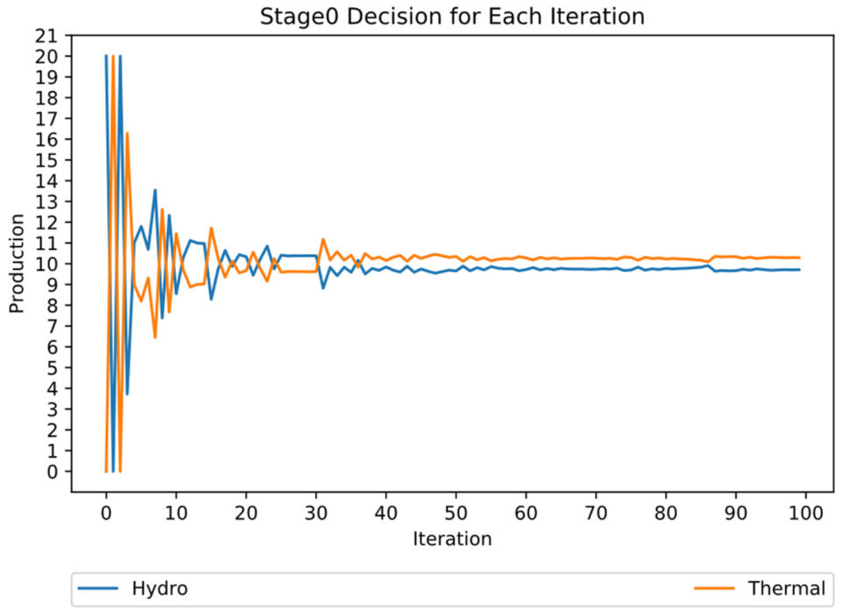
Hydro-thermal Energy Planning: Results

	Hydro Electricity Production	Thermal Electricity Production	Computation Time (s)
MSP	10.31	9.69	-
SDDP	9.84	10.16	1253
VFGL	10.67	9.33	1241

VFGL



SDDP



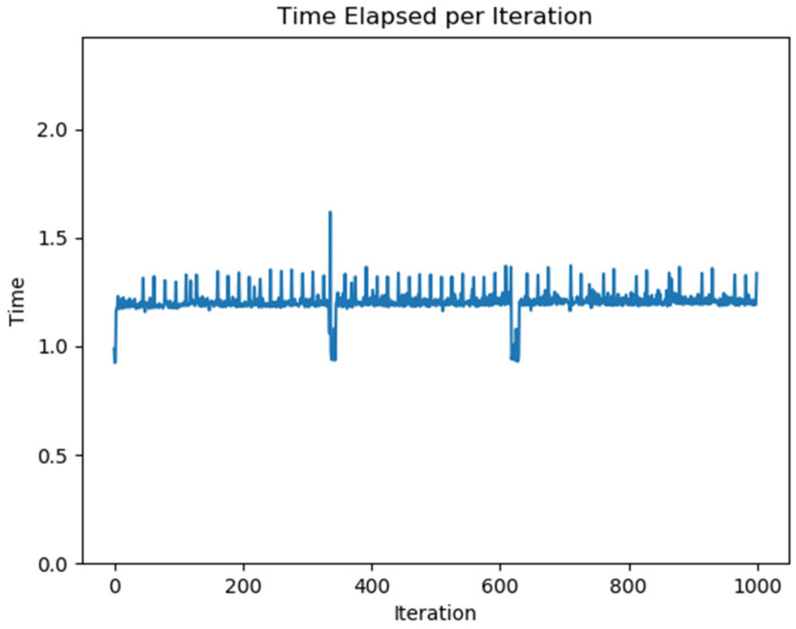
Hydro-thermal Energy Planning: Results

Parametric Function Form

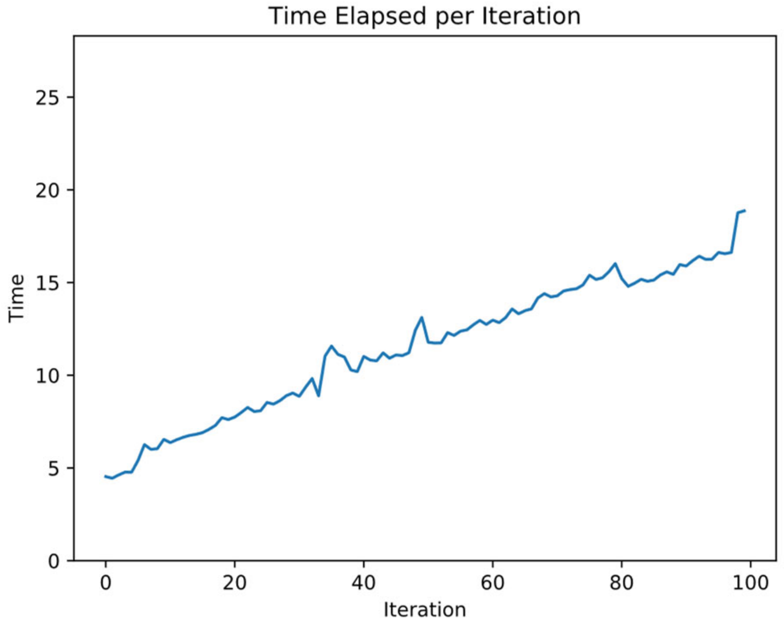
$$g_t\left(r_t|\theta_t = (\theta_{1,t}, \theta_{2,t})\right) = r_t \cdot \theta_{1,t} + e^{-\theta_{2,t} r_t + 5}$$

Trained value of $\theta_0 = [-4.2288, 1.0000]$

VFGL



SDDP



Hydro-thermal Energy Planning: Perturbation

Parameter		VFGL				MSP		
c_t^W	c_t^H	KKT deviation	Objective value	Hydro plant generation	Thermal plant generation	Objective value	Hydro plant generation	Thermal plant generation
2	5	2.24	365.00	5.06	14.94	370.74	5.39	14.61
2	5.5	2.68	374.01	6.26	13.74	378.38	7.53	12.47
2	6	2.45	382.01	7.68	12.32	378.99	8.36	11.64
2	6.5	3.24	389.62	8.91	11.09	392.04	9.8	10.2
*2	*7	3.66	396.64	9.90	10.10	400.39	10.45	9.55
2	7.5	3.6	402.68	10.91	9.09	404.15	10.88	9.12
2	8	4.52	408.99	11.61	8.39	417.47	13.18	6.82
3	7	2.66	521.85	7.67	12.33	523.37	8.68	11.32
3.5	7	2.44	583.90	6.49	13.51	585.67	6.89	13.11
4	7	1.87	645.08	4.88	15.12	644.56	5.11	14.89
4.5	7	1.89	705.74	2.78	17.22	711.22	3.91	16.09
5	7	1.35	764.90	0.81	19.19	765.76	1.54	18.46

Merton's Portfolio Optimization

Original Problem Formulation

$$\max \quad E\left[\int_0^T e^{-\rho t} U\big(C(t)\big)dt \; + \epsilon^{\gamma} e^{-\rho\, T} U(W_T)\right]$$

$$\text{where} \quad dW = [(w(t)(\mu - r) + r)W(t) - C(t)]dt + w(t)\sigma W(t) dB_t$$

$$U(x) = \begin{cases} \ln(x) & \text{if } \gamma = 1 \\ \frac{x^{1-\gamma}}{1-\gamma} & \text{if } \gamma \neq 1 \end{cases}$$

Known Analytical Solution

$$w^*(W,t) = \frac{\mu - r}{\sigma^2 \gamma}$$

$$C^*(W,t) = \begin{cases} \nu\big(1 + (\nu\epsilon - 1)e^{\nu(T-t)}\big)^{-1} W & \text{if } T < \infty \text{ and } \nu \neq 0 \\ (T - t + \epsilon)^{-1} W & \text{if } T < \infty \text{ and } \nu = 0 \\ \nu W & \text{if T} = \infty \end{cases}$$

where $\nu = \frac{\rho}{\gamma} - (1-\gamma)((\mu - r)w^*(W,t)\backslash 2\gamma + r/\gamma\ )$

Merton's Portfolio Optimization

Objective: Optimize asset allocation and consumption that maximizes total consumption utility and bequest utility.

Decision Variables	
C_t	Consumption at stage t
W_t	Wealth in the beginning of stage t
S_t	Amount invested into stock at stage t
B_t	Amount invested into bond at stage t

Parameter	
ρ	Discount rate
γ	Utility risk aversion coefficient
μ, σ	Mean return, volatility respectively, of stock
r	Risk free rate of bond
ϵ	Scaling coefficient of bequest utility
T	Time horizon
dt	Discretized time interval. T is a integer multiple of dt

Random Variable	
ξ_t	Increment of the Wiener process for dt

Merton's Portfolio Optimization: Problem Formulation

Stage 1

minimize	$-U(C_1)$	
subject to	$S_1 + B_1 + C_1 = 1$	Initial wealth
	$S_1, B_1, C_1 \geq 0$	Non-negativity

Stage t

minimize	$-U(C_t)$	
subject to	$S_t + B_t + C_t = rdtB_{t-1} + e^{\left(\mu-\frac{\sigma^2}{2}\right)\mathrm{dt}+\sigma\sqrt{\mathrm{dt}}*\xi} S_{t-1}$	Investment realization
	$S_t, B_t, C_t \geq 0$	Non-negativity

Stage T

minimize	$-U(W_T)$	
subject to	$W_T = rdtB_{T-1} + e^{\left(\mu-\frac{\sigma^2}{2}\right)\mathrm{dt}+\sigma\sqrt{\mathrm{dt}}*\xi} S_{T-1}$	Investment realization

Parametric Function Form

$$g_t\left(S_t, B_t | \theta_t = (\theta_{1,t}, \theta_{2,t}, \theta_{3,t}, \theta_{4,t})\right) = \sum_{i=1}^{n_t} -\theta_{1,t} ln(rB_t + \beta_{i,t} S_t) - \theta_{2,t} ln(\theta_{3,t} B_t + \theta_{4,t} S_t)$$

Where $\beta_{i,t}$ is a sampled value of $e^{\left(\mu-\frac{\sigma^2}{2}\right)\mathrm{dt}+\sigma\sqrt{\mathrm{dt}}*\xi_{i,t}}$

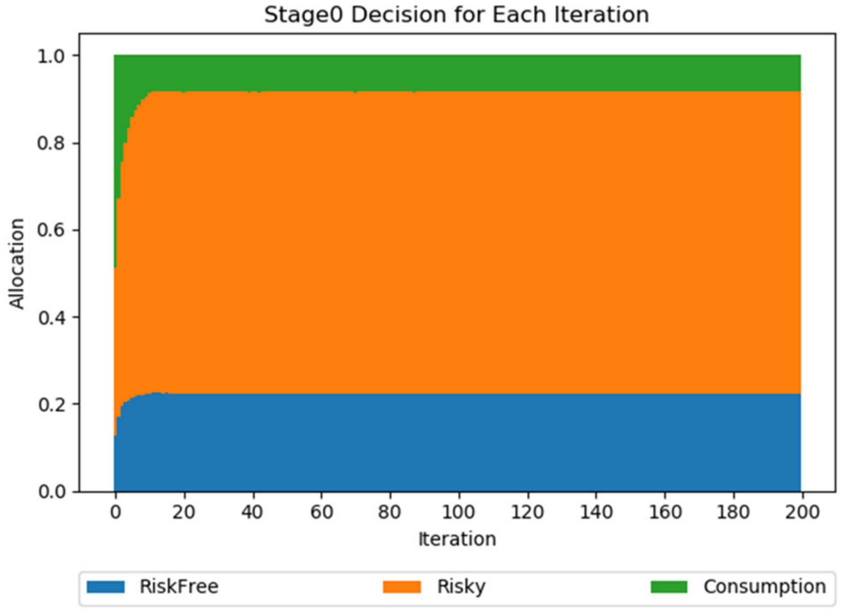
Initial value of $\theta_t = \left(\frac{1}{n_t}, \frac{1}{n_t}, 1, 1\right)$

$n_t = 20$

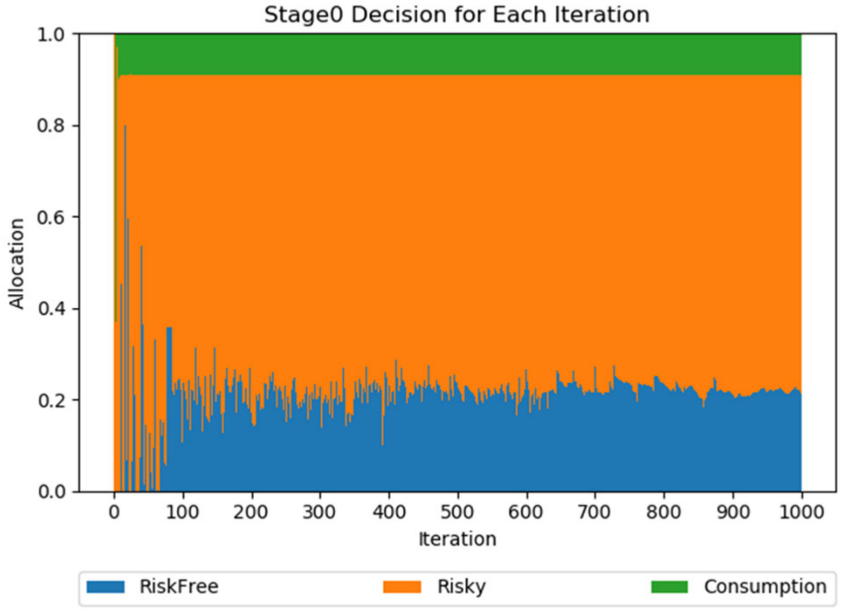
Merton's Portfolio Optimization: Results

	Risk Free Asset	Risky Asset	Consumption	Computation Time (s)
Analytical	0.2292	0.6875	0.0833	-
SDDP	0.2238	0.6907	0.0855	29087
VFGL	0.2214	0.6927	0.0834	677

VFGL



SDDP



Merton's Portfolio Optimization: Results

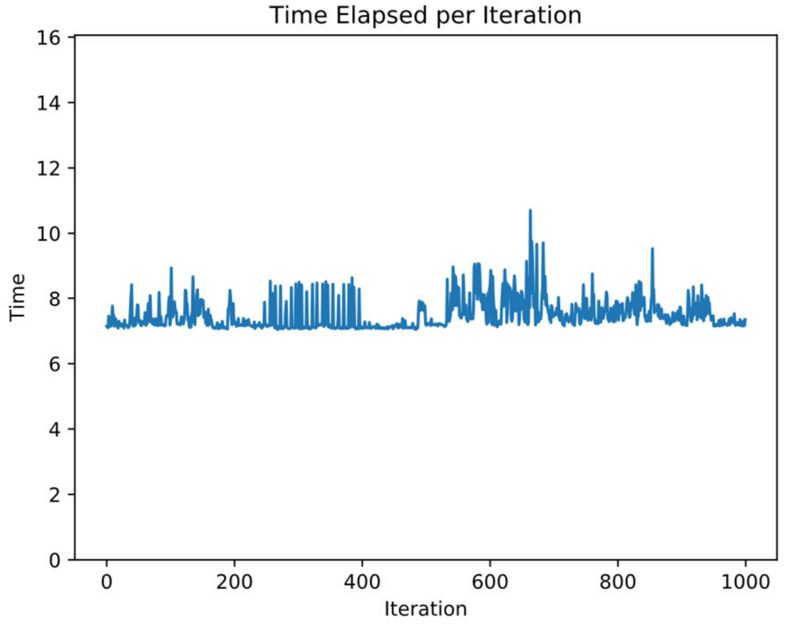
Parametric Function Form

$g_t\left(S_t, B_t | \theta_t = (\theta_{1,t}, \theta_{2,t}, \theta_{3,t}, \theta_{4,t})\right) = \sum_{i=1}^{n_t} -\theta_{1,t} ln(rB_t + \beta_{i,t} S_t) - \theta_{2,t} ln(\theta_{3,t} B_t + \theta_{4,t} S_t)$

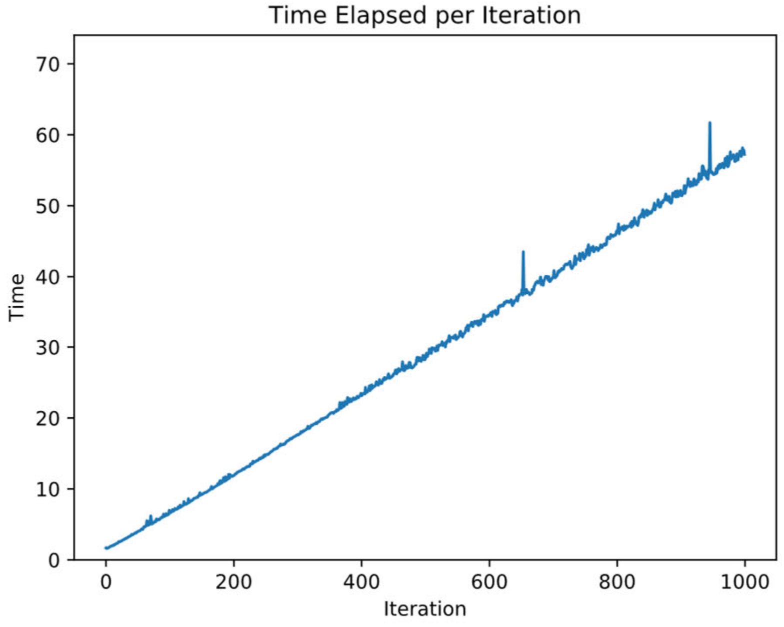
Where $\beta_{i,t}$ is a sampled value of $e^{\left(\mu - \frac{\sigma^2}{2}\right)\mathrm{dt} + \sigma\sqrt{\mathrm{dt}} * \xi_{i,t}}$

Trained value of θ_0 = [0.54287, 0.0747, 0.9986, 1.0014]

VFGL



SDDP



Merton's Portfolio Optimization: Perturbation

Parameter			**VFGL**					**Analytical**			
Risk-free return	Risky return	Risky volatility	KKT deviation	Objective	Risk free	Risky	Consumption	Objective	Risk free	Risky	Consumption
0.04	0.060	0.20	0.0114	29.5281	0.4567	0.4600	0.0833	29.5266	0.4564	0.4603	0.0833
0.034	0.060	0.26	0.0105	29.5245	0.5122	0.4045	0.0833	29.5458	0.5120	0.4046	0.0833
0.031	0.060	0.29	0.0107	29.5399	0.5367	0.3799	0.0833	29.5565	0.5372	0.3795	0.0833
*0.030	*0.060	*0.20	0.0075	29.5112	0.2284	0.6883	0.0833	29.5508	0.6885	0.2282	0.0833
0.028	0.060	0.32	0.0106	29.5429	0.5571	0.3596	0.0833	29.5667	0.5580	0.3587	0.0833
0.025	0.060	0.35	0.0108	29.5567	0.5746	0.3421	0.0833	29.5768	0.5755	0.3412	0.0833
0.022	0.060	0.38	0.0109	29.5703	0.5900	0.3267	0.0833	29.5867	0.5910	0.3257	0.0833
0.019	0.060	0.41	0.0110	29.5791	0.6036	0.3130	0.0833	29.5966	0.6047	0.3119	0.0833
0.016	0.060	0.44	0.0110	29.5905	0.6159	0.3008	0.0833	29.6065	0.6171	0.2996	0.0833
0.013	0.060	0.47	0.0111	29.5993	0.6270	0.2897	0.0833	29.6163	0.6282	0.2884	0.0833
0.01	0.060	0.5	0.0112	29.6081	0.6371	0.2795	0.0833	29.6261	0.6384	0.2783	0.0833

Comparison to the Conventional Algorithms: Advantage

	SDDP	VFGL
Computational Burden Every Iteration	Increases every iteration	Remains constant
Scenario tree approximation	Required	Not required, but can employ them for a minibatch sampling
Parameter recycling (transfer learning)	Not possible at all	Limited possibility by sharing the same parametric form or/and initial parameter values (computational potential in solving many perturbed problems)
Online/Offline	Offline learning. Parallel computation is difficult	Online learning. Parallel computation is easy

Comparison to the Conventional Algorithms: Disadvantage

	SDDP	VFGL
Convergence to the optimal	Guaranteed for a given scenario tree	Not guaranteed. Optimality only indirectly checked by the inspection of solution and KKT deviation
Required user input	Few parameters about iteration and stopping criterion needed. They do not affect the solution quality much	Parametric form of value function and initial values need, which are critical for the performance
Optimality Gap	Estimate of optimality gap can be computed for a given scenario tree	Only the upper bound of the expected objective value available

Comparison to the Conventional Algorithms

	SDDP	VFGL
Stagewise independence of stochastic process assumption	Assumed. Can be extended to the HMM scheme.	Assumed. Can be extended to the HMM scheme.
Relatively complete recourse assumption	Assumed. Feasibility cuts can relax the assumption.	Assumed. Feasibility cuts can relax the assumption.
Additive Separation of current and past stage decision variables	Assumed.	Assumed.

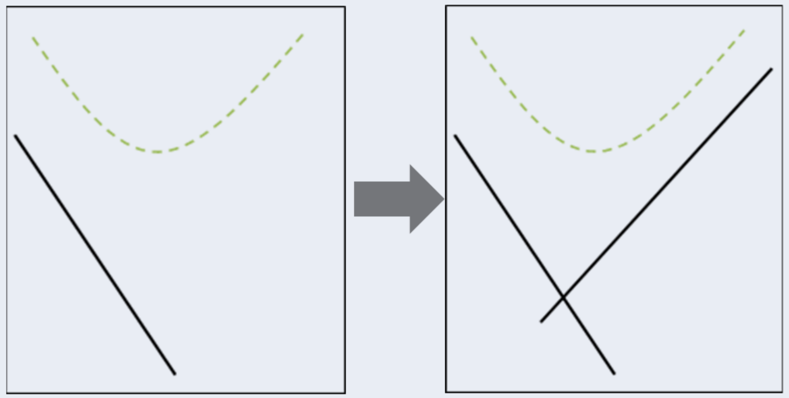
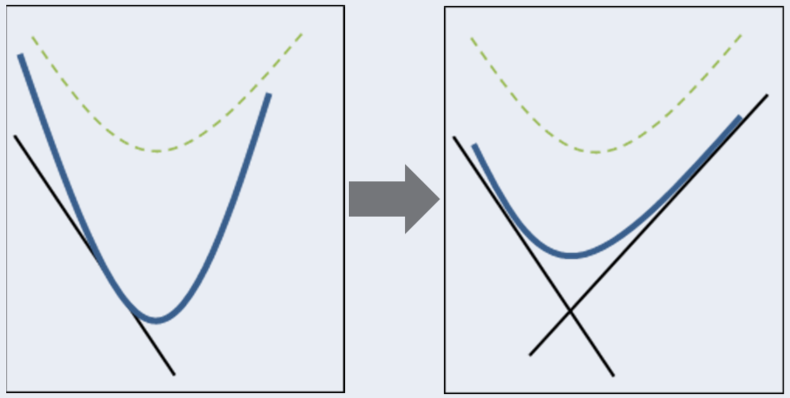
Part 2-2

DEEP VALUE FUNCTION NETWORKS FOR LARGE-SCALE STOCHASTIC OPTIMIZATION PROGRAMS

Based on:
Hyunglip Bae, Jinkyu Lee, Woo Chang Kim, Yongjae Lee, 2023, Deep value function networks for large-scale multistage stochastic programs, *AISTATS 2023*, Accepted

Value Function Gradient Learning (VFGL)

- **Value function is approximated by a fixed parametric families**
  - Learn the parameter that well approximates the gradient of value function
  - No extra constraint (optimality cut) is added to the problem

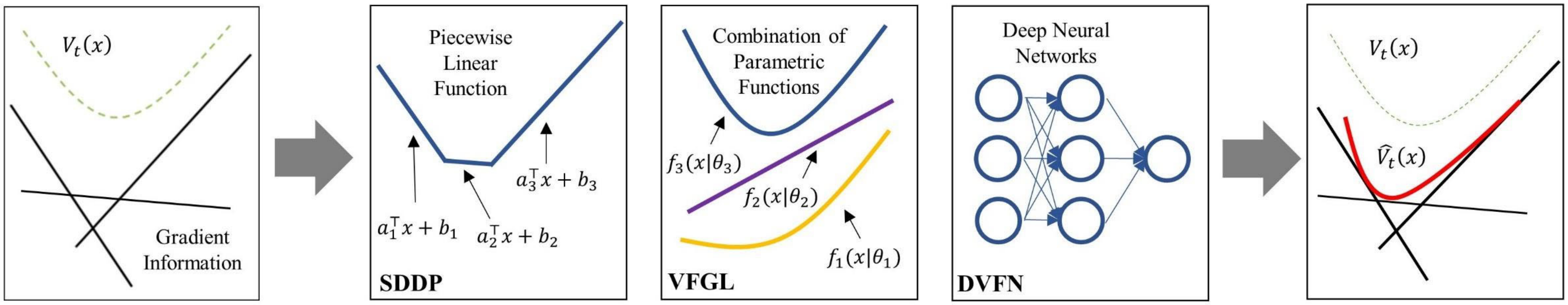
Piecewise-linear approximation on the value function	VFGL
$\widehat{V}_t(x) = \max\{a_i^{\mathrm{T}} x + b_i\}_i$	$\widehat{V}_t(x) = g_t(x \mid \theta_t)$
	

- **The performance of VFGL is heavily dependent upon the choice of parametric form, and it affects the convergence to the optimality**

Deep Value Function Networks (DVFN)

- **Value function is approximated by input convex neural networks**
  - Learn the parameter that well approximates the gradient of value function
  - No extra constraint (optimality cut) is added to the problem
  - Given neural networks' extreme capability in function approximation, users do not need an artistic sense of choosing the appropriate functional form

- **Construction of approximation in each algorithm**



$\widehat{V}_t(x) = max\{a_i^\top x + b_i\}$ $\widehat{V}_t(x) = \sum f_i(x|\theta_i)$ $\widehat{V}_t(x) = g(W_i^{(x)} x + b_i)$

Problem Definition: Stagewise Decomposition

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

$$V_{t+1}(x_t | \xi_{[t]}) \coloneqq \mathbb{E}_{\cdot|\xi_{[t]}}[\mathcal{V}_{t+1}(x_t, \xi_{[t+1]})]$$

For t = 1

$$\begin{array}{lll} \textit{Minimize}_{x_1} & f_1(x_1) + V_2(x_1) & \\ \textit{Subject to} & g_{1,i}(x_1) \le -h_{1,i} & i = 1, \dots, p_1 \\ & l_{1,j}(x_1) = b_{1,j} & j = 1, \dots, q_1 \end{array}$$

For $t = 2, \dots, T-1$

$$\begin{array}{lll} \textit{Minimize}_{x_t} & f_t(x_t, \xi_t) + V_{t+1}(x_t | \xi_{[t]}) & \\ \textit{Subject to} & g_{t,i}(x_t, \xi_t) \le -h_{t,i}(x_{t-1}, \xi_t) & i = 1, \dots, p_t \\ & l_{t,j}(x_t, \xi_t) = b_{t,j}(x_{t-1}, \xi_t) & j = 1, \dots, q_t \end{array}$$

f: convex
g, h: twice-differentiable convex
l, b: linear

For $t = T$

$$\begin{array}{lll} \textit{Minimize}_{x_T} & f_T(x_T, \xi_T) & \\ \textit{Subject to} & g_{T,i}(x_t, \xi_t) \le -h_{T,i}(x_{T-1}, \xi_T) & i = 1, \dots, p_T \\ & l_{T,j}(x_t) = b_{T,j}(x_{T-1}, \xi_T) & j = 1, \dots, q_T \end{array}$$

DVFN Loss function construction: KKT condition

Stationarity:

$$\boxed{\nabla f_t(x_t,\xi_t)+\nabla\hat{V}_{t+1}(x_t|\theta_{t+1})+\textstyle\sum_{i=1}^{k}u_{t,i}\nabla g_{t,i}(x_t,\xi_t)+\sum_{i=1}^{p}v_{t,i}\nabla l_{t,j}(x_t,\xi_t)=0}$$

Primal Feasibility:

$$g_{t,i}(x_t,\xi_t)\le -h_{t,i}(x_{t-1},\xi_t)\qquad i=1,\dots,k_t$$

$$l_{t,j}(x_t,\xi_t)=b_{t,j}(x_{t-1},\xi_t)\qquad j=1,\dots,p_t$$

Dual Feasibility:

$$u_{t,i}\ge 0,\ i=1,\dots,p_t$$

Complementary Slackness:

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

DVFN Loss function construction: KKT condition

Stationarity:

$$\nabla f_t(x_t,\xi_t) + \nabla V_{t+1}(x_t) + \sum_{i=1}^{k} u_{t,i}\nabla g_{t,i}(x_t,\xi_t) + \sum_{i=1}^{p} v_{t,i}\nabla l_{t,j}(x_t,\xi_t) = \nabla V_{t+1}(x_t) - \nabla \hat{V}_{t+1}(x_t|\theta_{t+1})$$

Primal Feasibility:

$$g_{t,i}(x_t,\xi_t) \le -h_{t,i}(x_{t-1},\xi_t) \quad i = 1,\dots,k_t$$

$$l_{t,j}(x_t,\xi_t) = b_{t,j}(x_{t-1},\xi_t) \quad j = 1,\dots,p_t$$

Closer the $\nabla V_{t+1}(x_t) - \nabla \hat{V}_{t+1}(x_t|\theta_{t+1})$ is to 0, better the approximation $\hat{V}_{t+1}(x_t|\theta_{t+1})$

$\downarrow$

DVFN minimizes the loss $\left\|\nabla V_{t+1}(x_t) - \nabla \hat{V}_{t+1}(x_t|\theta_{t+1})\right\|^2$

Dual Feasibility:

$$u_{t,i} \ge 0,\ i = 1,\dots,p_t$$

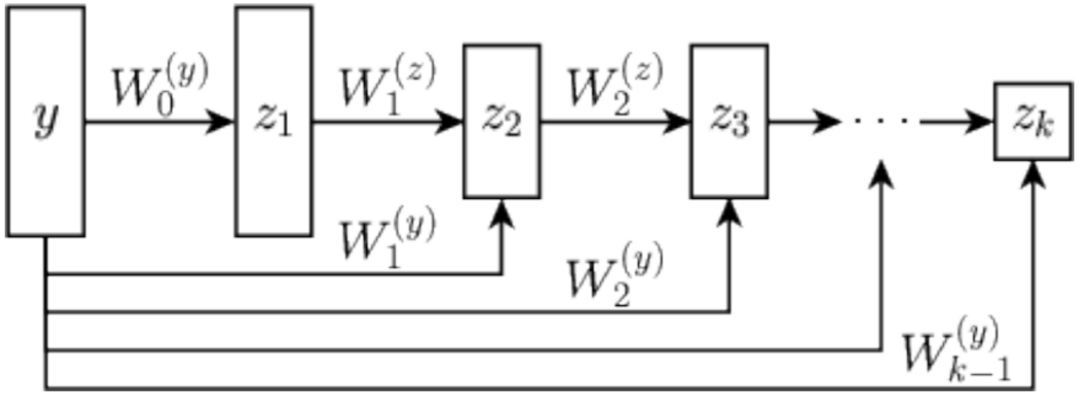
Complementary Slackness:

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

Input Convex Neural Networks

- **Neural networks architecture for approximation of the value function**
  - The value function is convex under mild conditions
  - For stability of DVFN, it is necessary to keep the neural networks convex to the input
  - We stabilize DVFN algorithm by using Input Convex Neural Networks (ICNN) that have special network architecture to ensure convexity with respect to inputs

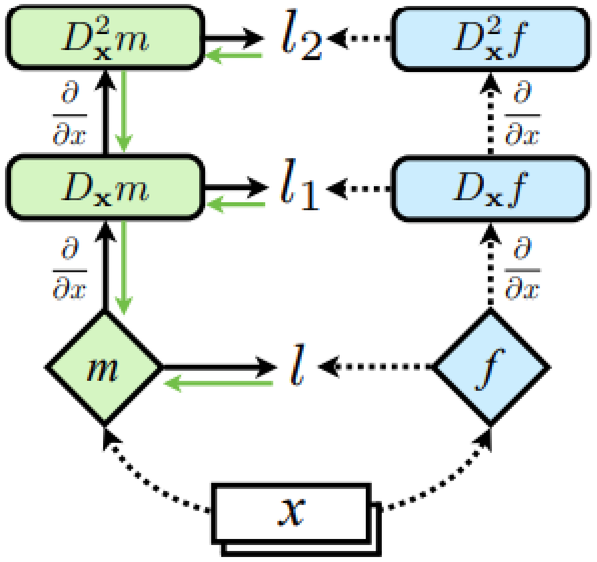
- **Structure of ICNN**



Sobolev training

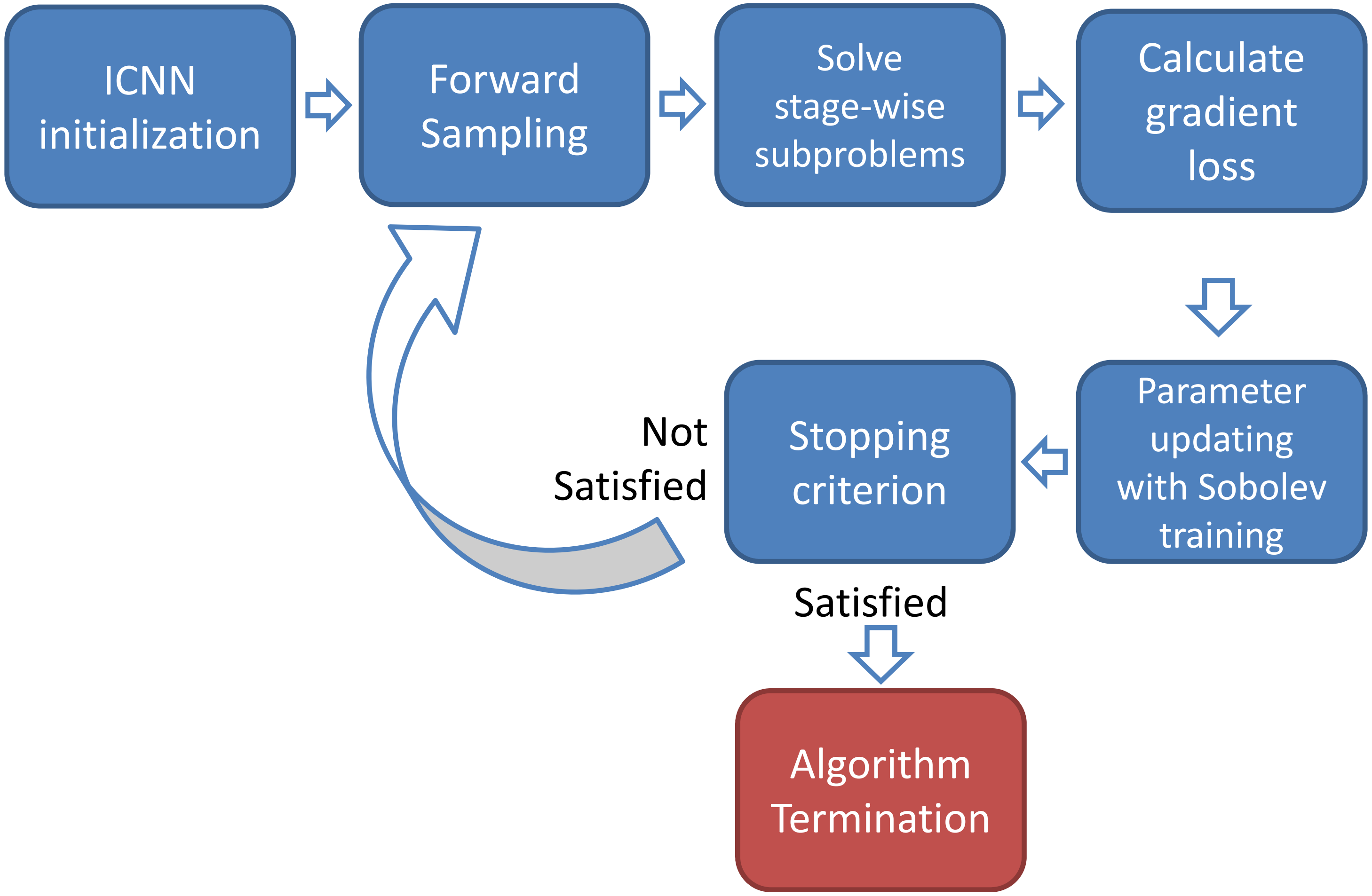
- **Sobolev training for gradient loss function**
  - The loss function for Neural Networks is usually designed to make the neural networks to output a value that is close to the value of the target function
  - DVFN approximates value functions based on their gradient information, not outputs, so the loss function should incorporate the difference between the gradient of ICNN
  - We adopted the Sobolev training, which approximates not only the output of the function but also its gradient

- **Framework of Sobolev training**



a)

DVFN Framework Summary



Numerical Experiments

- **Two popular problems in optimization community are considered**
  - Production optimization
  - Energy planning

- **Each problem is solved using four different approaches**
  - MSP
  - SDDP
  - VFGL
  - DVFN

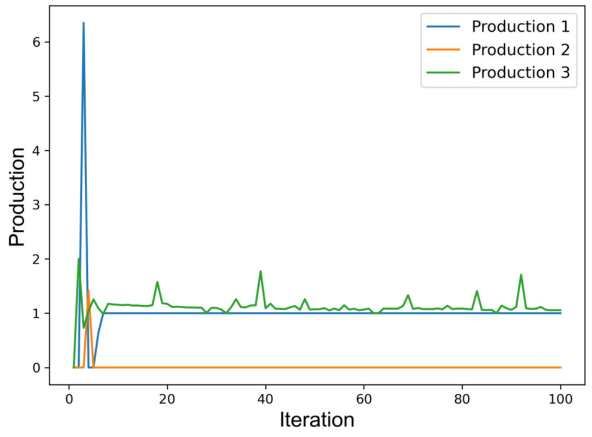
- **For VFGL, we use three different types of parametric forms**
  - Exponential (VFGLexp)
  - Quadratic (VFGLquad)
  - Linear (VFGLlinear)

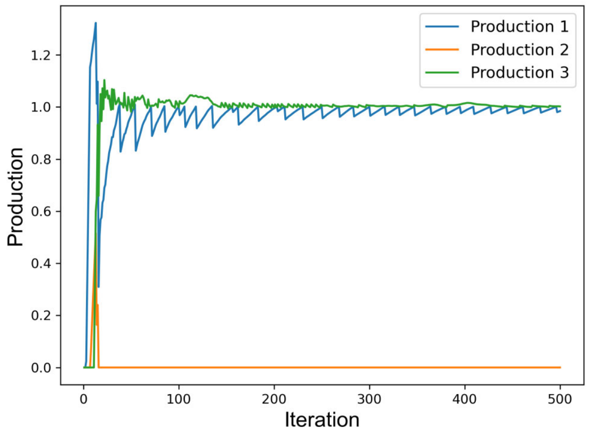
Production Optimization: Results

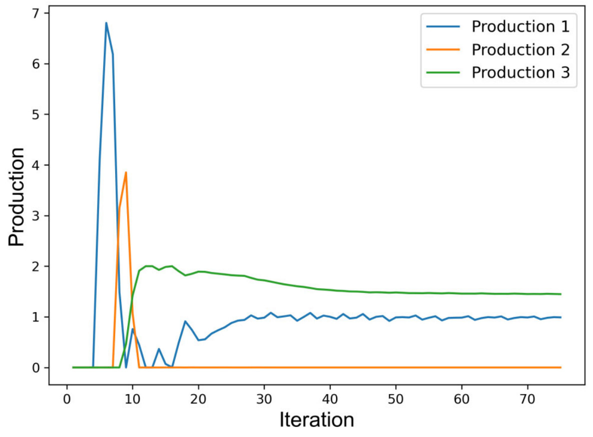
- **Performance comparisons of algorithm**

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)

- **First stage decisions of algorithms for each iteration**





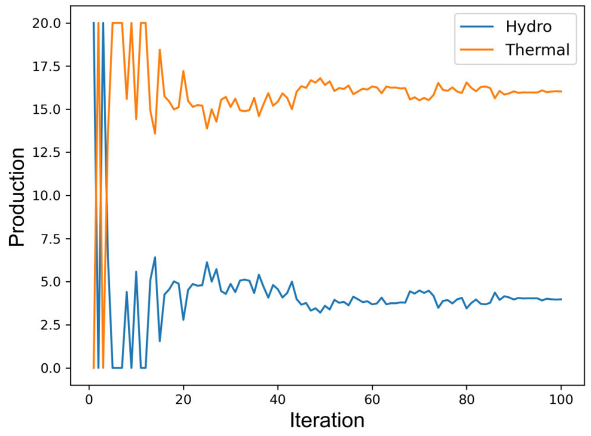


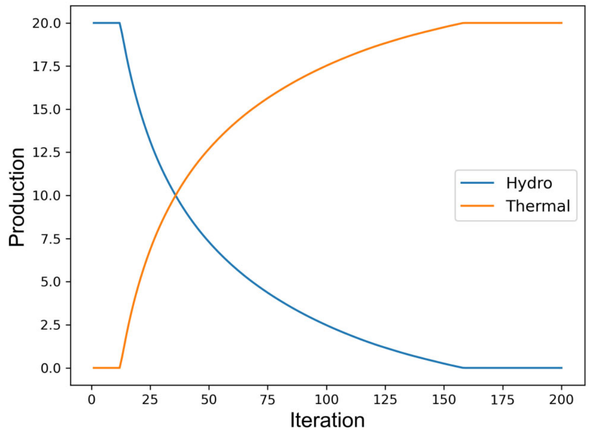
Energy Planning: Results

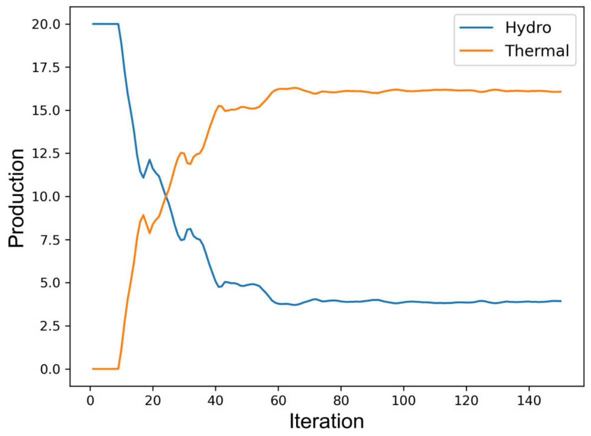
- **Performance comparisons of algorithm**

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)

- **First stage decisions of algorithms for each iteration**

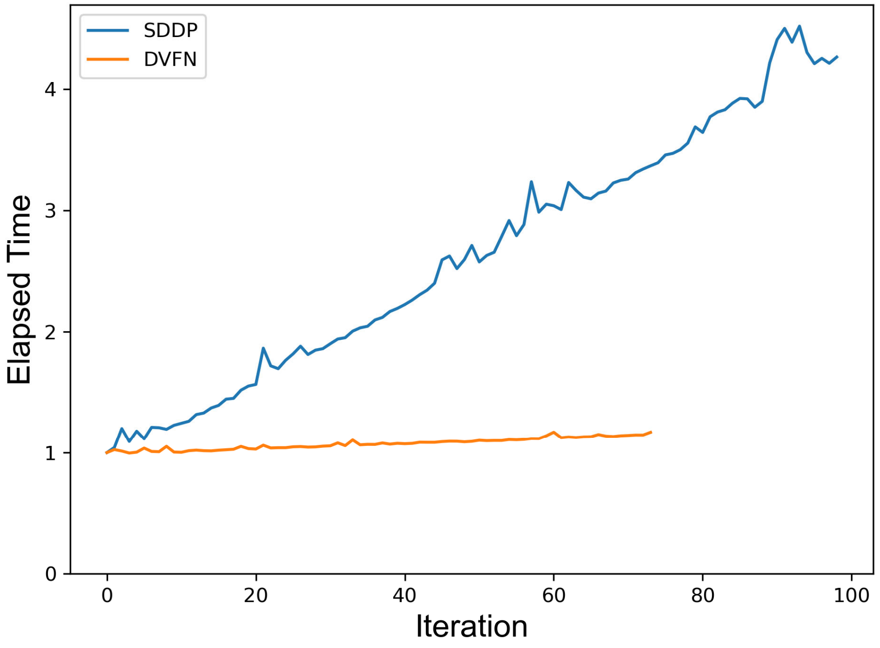




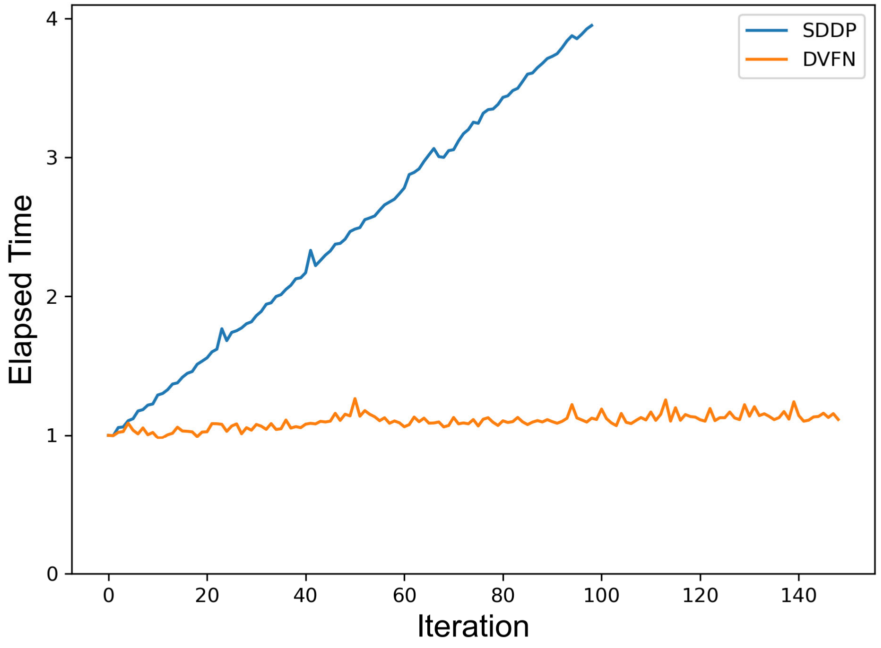


Efficiency of DVFN: Time elapsed per iteration

- **Portfolio optimization**



- **Energy planning**



Part 2-3

TRANSFORMER-BASED STAGEWISE DECOMPOSITION FOR LARGE-SCALE MULTISTAGE STOCHASTIC OPTIMIZATION

Based on:
Chan Yeong Kim, Jongwoong Park, Hyunglip Bae, Woo Chang Kim, 2023, Transformer-based Stagewise Decomposition for Large-Scale Multistage Stochastic Optimization, *ICML 2023*, Submitted

Improving SDDP: Generation of Cuts

- Dai et al. (2021) proposed **$\boldsymbol{\nu}$-SDDP.**

- In ν-SDDP, neural network is trained using a meta-learning approach to **generate a fixed number of corresponding cuts** based on the problem context vector.

- This is the novel algorithm that generates cuts using neural network.

- Drawbacks:
  - Previously generated cuts are not considered to generate new ones.
  - Only linear programs can be solved by ν-SDDP.
  - The number of cuts to be generated is fixed.

Motivation

- In SDDP, calculation of new cuts **depends on** previously generated cuts.
- Cuts are generated **sequentially**
- Until the convergence condition is satisfied.
  - **Unable to specify** the number of cuts to be generated.

$\Downarrow$

How to apply a **sequence model** to the process in which cuts are generated sequentially?

- We propose a model, ***TranSDDP***, that uses architecture of **Transformer** to generate the piecewise linear function for approximating the value function in SDDP

Methodology: Input and Output Sequence

[Input Sequence]

$$\left\{\left(\lambda_{A_{t,i}}^{1}, \dots, \lambda_{A_{t,i}}^{M_{A_{t,i}}}, \lambda_{B_{t,i}}^{1}, \dots, \lambda_{B_{t,i}}^{M_{B_{t,i}}}, \lambda_{b_{t,i}}^{1}, \dots, \lambda_{b_{t,i}}^{M_{b_{t,i}}}, \tilde{t}\right)_{i=1}^{N}\right\} \; for \; t = 1, \dots, T-1$$

- Parameters of probability distribution for A_t, B_t, b_t of feasible region.
- Subscript is coefficient of the i-th constraint of the feasible region at stage t.
- Superscript is the j-th of M_c parameters of the distribution of the corresponding coefficient c.
- Assume λ is sampled from the prior distribution to consider the problem defined with a parametric family.
  - $c \sim P_c\left(\cdot \,\middle|\, \lambda_c^1, \dots, \lambda_c^{M_c}\right) \; and \; \lambda_c^j \sim P_{\lambda_c^j}(\cdot)$
- $\tilde{t} \coloneqq t/(T-1)$

Methodology: Input and Output Sequence

[Output Sequence]

$$\{\tilde{\beta}_k, \tilde{\alpha}_k, \tilde{\tau}_k\}_{k=1}^{K}$$

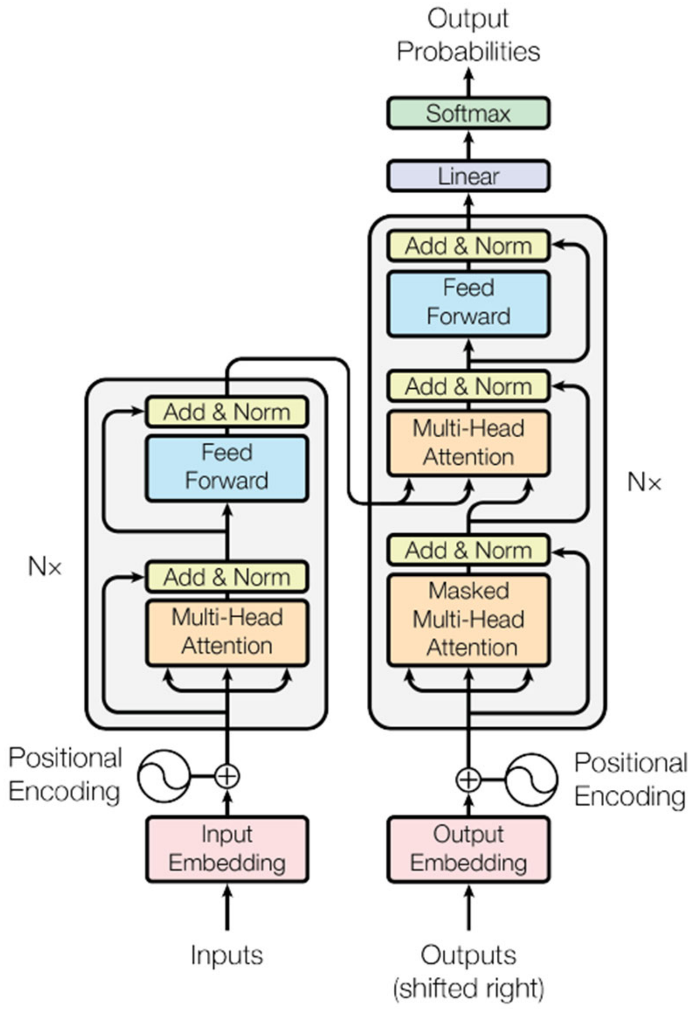
- $\tilde{\beta}_k \in \mathbb{R}^d$ *and* $\tilde{\alpha}_k \in \mathbb{R}$ are the gradient and intersection of the kth cut.
  - d: dimension of decision variables.

- τ_k: token
  - One hot encoded category information
  - $(1, 0, 0, 0)$: padding
  - $(0, 1, 0, 0)$: start
  - $(0, 0, 1, 0)$: middle
  - $(0, 0, 0, 1)$: end

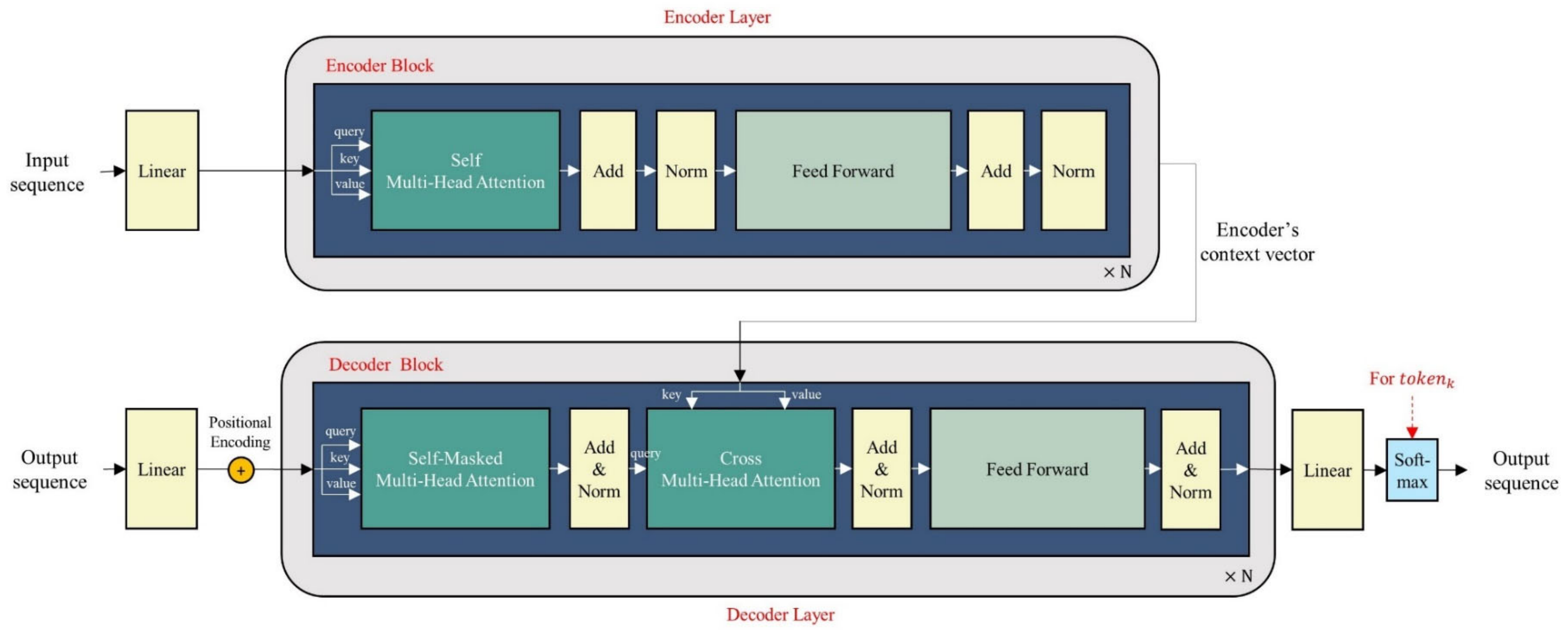
- TranSDDP generates a sequence of $\tilde{\beta}_k, \tilde{\alpha}_k$ until token indicates the end.
  - Then, construct the approximated value function
  - $\tilde{Q}_t(x_{t-1})$: $\max_{k=1,\ldots,K}\{(\tilde{\beta}_k)^\top x_{t-1} + \tilde{\alpha}_k\}$.

Methodology: Model Architecture

- **TranSDDP** mainly follows the architecture of the Transformer except a few adjustments.



Methodology: Model Architecture – TranSDDP



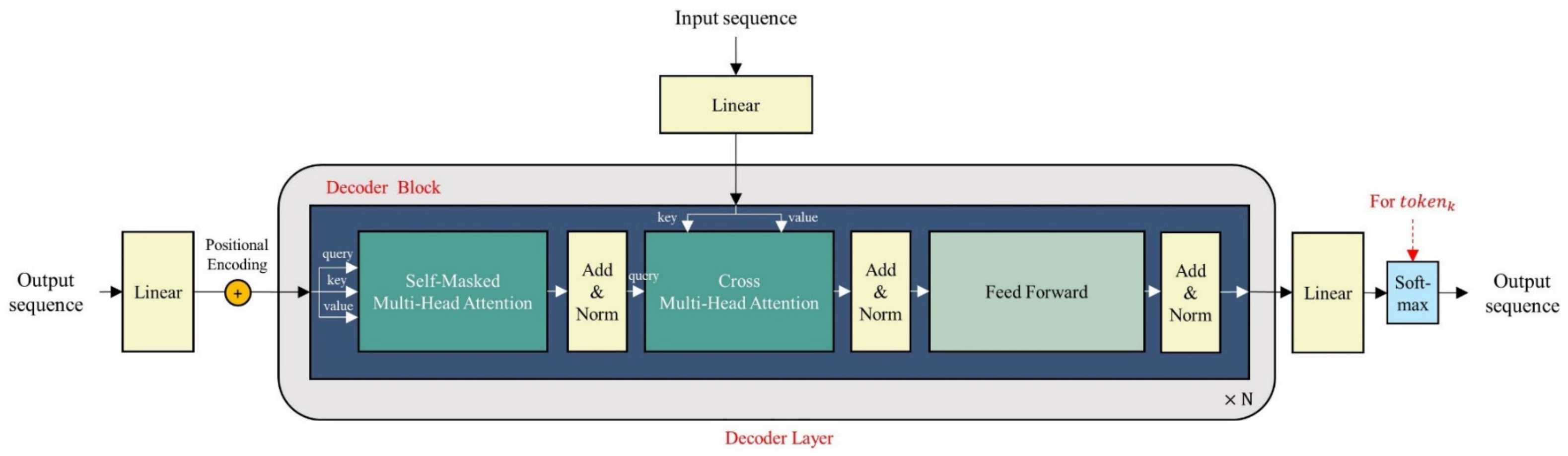
- Replace the input and output embedding layer with a linear layer.
  - Input and output sequence are continuous.
- Output layer of Transformer, softmax layer, is applied only for the vectors that pertain to τ_k.
- Add positional encoding only to the output sequence.
  - Positional information of the input sequence is not crucial.

Methodology: Model Architecture – TranSDDP-Decoder

- Herein, it is noted that:
  - The size of the input sequence is fixed.
  - The importance of the relationship between input sequence is relatively insignificant.

- Self-attention layer on the input sequence can have an adverse effect by increasing computational complexity without much gain in performance.

- We propose **TranSDDP-Decoder**, which **employs solely the decoder component** of the TranSDDP model



Methodology: Learning System

[Dataset $\mathcal{D}_s$]

$$\mathcal{D}_s \coloneqq \left\{ z_s \coloneqq \left(\Lambda, \tilde{t}, \{\beta_k, \alpha_k, \tau_k\}_{k=1}^{K} \right)_s \right\}_{s=1}^{S}$$

- Λ: sampled parameters of the probability distribution for stochastic elements (A_t, B_t, b_t).
  - $\Lambda = \left\{\lambda^j\right\}_{j=1}^{M}$ $where\ \lambda^j \sim P_{\lambda^j}(\cdot)$

- Get $\{\beta_k, \alpha_k, \tau_k\}$ by solving the defined problem with SDDP.

Methodology: Learning System

[Loss Function]

$$L(W) \coloneqq \frac{1}{B}\frac{1}{K}\sum_{b=1}^{B}\sum_{k=2}^{K}\left[\left|\left|\beta_k - \tilde{\beta}_k\right|\right|_2^2 + (\alpha_k - \tilde{\alpha}_k) - \sum_{c=1}^{4}\tau_{k,c}\log(\tilde{\tau}_{k,c})\right]$$

- Based on the given dataset $\mathcal{D}_s$, the model parameters W are optimized by minimizing a loss function.
- Loss function = MSE Loss + CE Loss between the target output sequence $\{\beta_k, \alpha_k, \tau_k\}$ and the predicted output sequence $\{\tilde{\beta}_k, \tilde{\alpha}_k, \tilde{\tau}_k\}$.

Methodology: Learning System – more tricks

- The decoder takes the target output sequence ranging from $k = 1$ to $K - 1$.

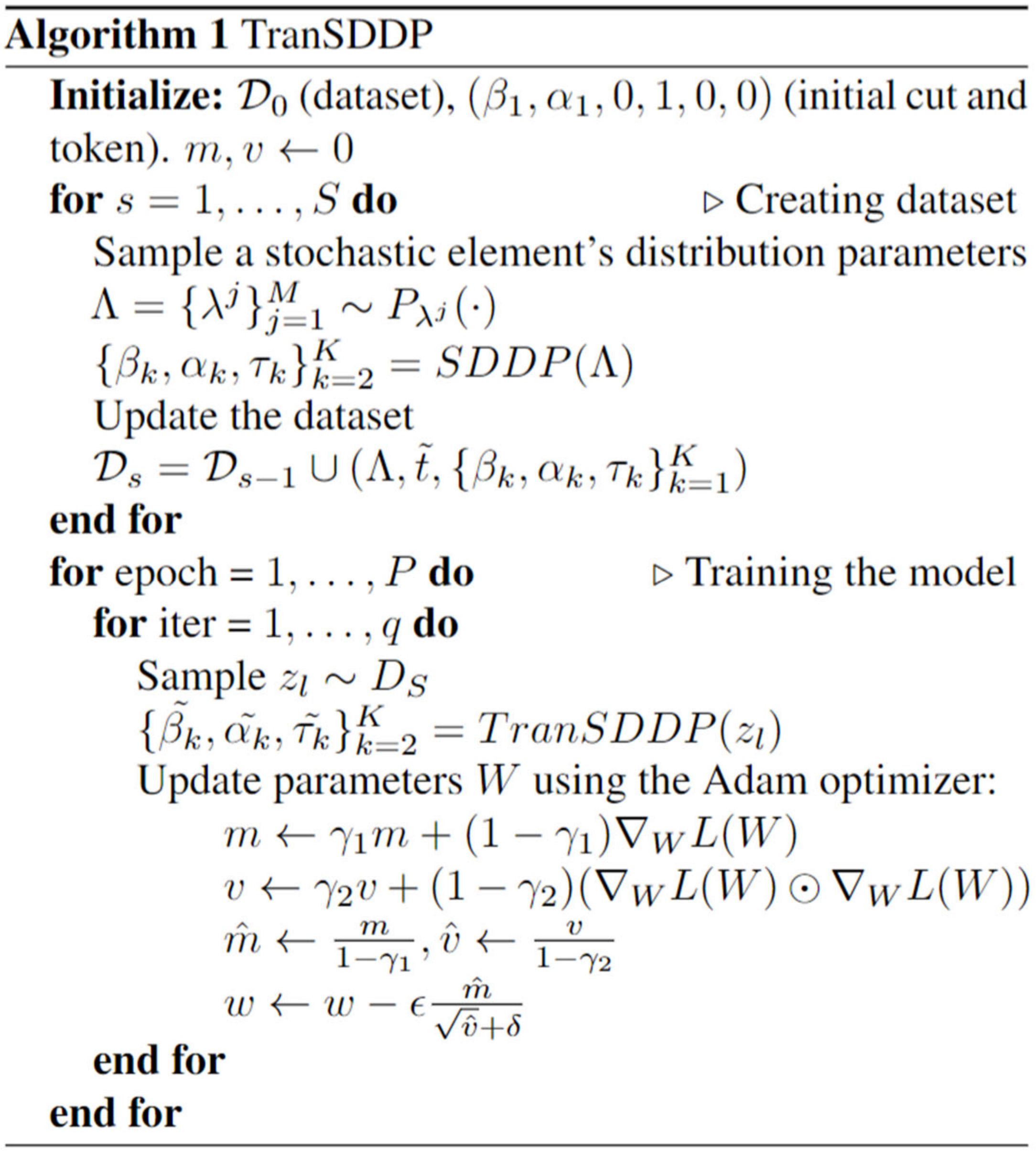
- Then, TranSDDP model outputs the sequence of predicted cuts and tokens ranging from $k = 2$ to K.

- By feeding the decoder with the target sequence instead the predicted sequence, we employ Teacher forcing.

- In this process, the 1^{st} cut is initialized arbitrarily to serve as a substitute for the starting token of the decoder.

- Similar to when actually implementing the SDDP algorithm, cuts such as $x > 0$ that provide minimal guidelines for approximating the value function can be used for initialization.

Methodology: Algorithm



Algorithm 1 TranSDDP

```
Initialize: $\mathcal{D}_0$ (dataset), $(\beta_1, \alpha_1, 0, 1, 0, 0)$ (initial cut and token). $m, v \leftarrow 0$
for $s = 1, \ldots, S$ do                                  ▷ Creating dataset
    Sample a stochastic element's distribution parameters
    $\Lambda = \{\lambda^j\}_{j=1}^{M} \sim P_{\lambda^j}(\cdot)$
    $\{\beta_k, \alpha_k, \tau_k\}_{k=2}^{K} = SDDP(\Lambda)$
    Update the dataset
    $\mathcal{D}_s = \mathcal{D}_{s-1} \cup (\Lambda, \tilde{t}, \{\beta_k, \alpha_k, \tau_k\}_{k=1}^{K})$
end for
for epoch = $1, \ldots, P$ do                              ▷ Training the model
    for iter = $1, \ldots, q$ do
        Sample $z_l \sim D_S$
        $\{\tilde{\beta}_k, \tilde{\alpha_k}, \tilde{\tau_k}\}_{k=2}^{K} = TranSDDP(z_l)$
        Update parameters $W$ using the Adam optimizer:
            $m \leftarrow \gamma_1 m + (1 - \gamma_1)\nabla_W L(W)$
            $v \leftarrow \gamma_2 v + (1 - \gamma_2)(\nabla_W L(W) \odot \nabla_W L(W))$
            $\hat{m} \leftarrow \frac{m}{1-\gamma_1}, \hat{v} \leftarrow \frac{v}{1-\gamma_2}$
            $w \leftarrow w - \epsilon \frac{\hat{m}}{\sqrt{\hat{v}}+\delta}$
    end for
end for
```

Experiments

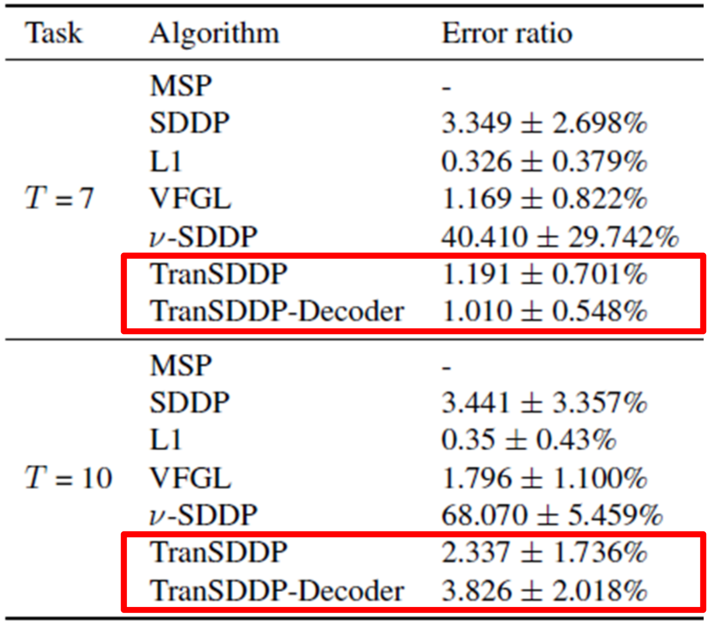
- Tasks
  - Task 1: Production Planning
  - Task 2: Energy Planning
  - Task 3: Lifetime Financial Planning
  - 7- and 10-stage MSP problems with 5 and 3 scenario branches

- Benchmarks
  - MSP, SDDP, L1-Dominance, VFGL, Neural SDDP

- Metric: averaging from 100 repeated experiments
  - Solution Quality
    - Error ratio $= \left|\frac{MSP's\ objective\ value - Candidate's\ objective\ value}{MSP's\ objective\ value}\right|$
  - Computational time
  - Infeasibility Test
  - Comparison of generated cuts

Experiments: Task 1 – Energy Planning

- Performance comparison of algorithms for energy planning

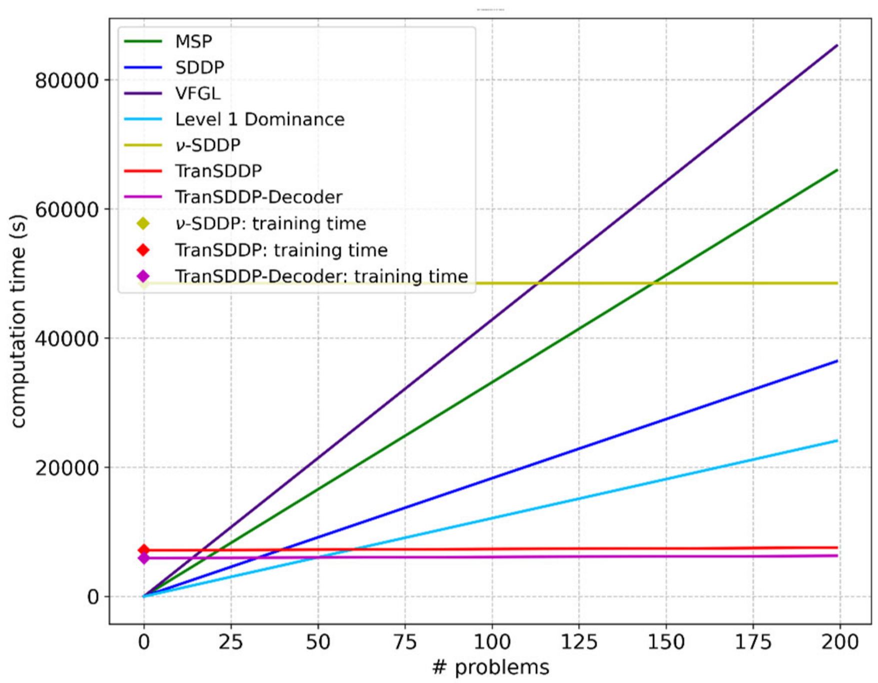


Task	Algorithm	Error ratio
$T=7$	MSP	-
	SDDP	$3.349 \pm 2.698\%$
	L1	$0.326 \pm 0.379\%$
	VFGL	$1.169 \pm 0.822\%$
	ν-SDDP	$40.410 \pm 29.742\%$
	TranSDDP	$1.191 \pm 0.701\%$
	TranSDDP-Decoder	$1.010 \pm 0.548\%$
$T=10$	MSP	-
	SDDP	$3.441 \pm 3.357\%$
	L1	$0.35 \pm 0.43\%$
	VFGL	$1.796 \pm 1.100\%$
	ν-SDDP	$68.070 \pm 5.459\%$
	TranSDDP	$2.337 \pm 1.736\%$
	TranSDDP-Decoder	$3.826 \pm 2.018\%$

- Higher error ratio compared to the L1 algorithm and VFGL
- Outperform other algorithms
- Note that: ν-SDDP has poor performance
  - ν-SDDP can only be applied to linear programs
  - Energy planning problems are not the form of a linear program
- Stable computation time with respect to the number of stages

Experiments: Task 1 – Energy Planning

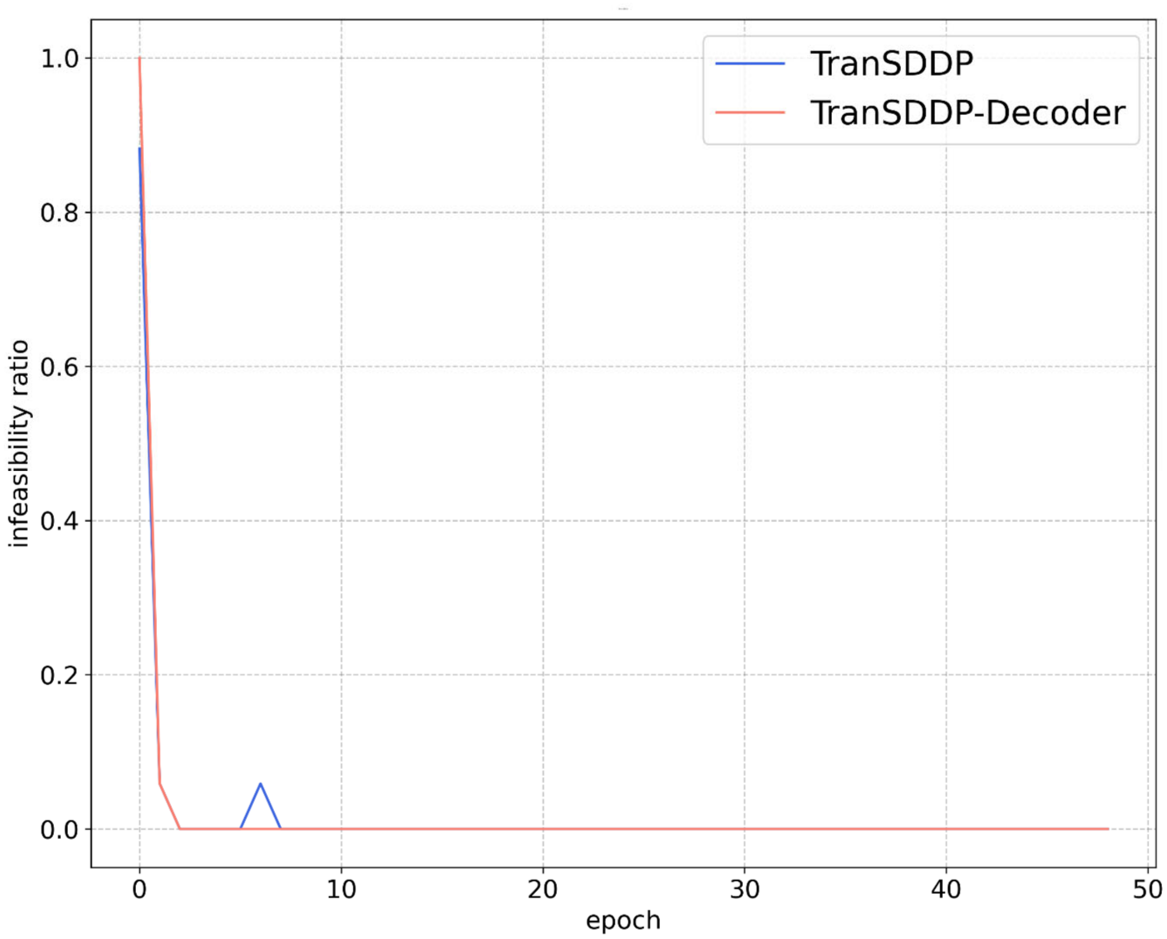
- Computation time: training time + evaluation time



  - The computation time of TranSDDP and TranSDDP-Decoder are almost constant
  - TranSDDP (TranSDDP-Decoder) has a computational advantage over SDDP for 39 (33) or more problems.

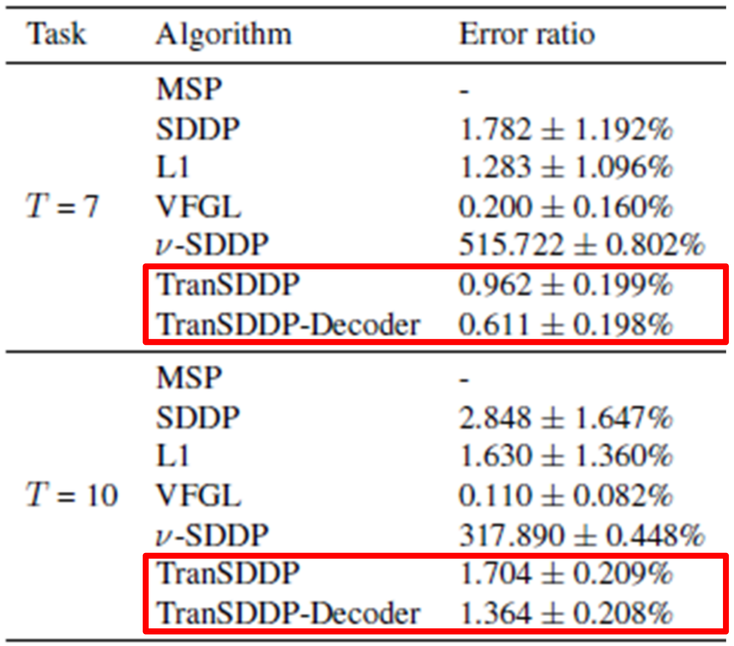
Experiments: Task 1 – Energy Planning

- Infeasibility test



Experiments: Task 2 – Financial Planning

- Performance comparison of algorithms for lifetime financial planning

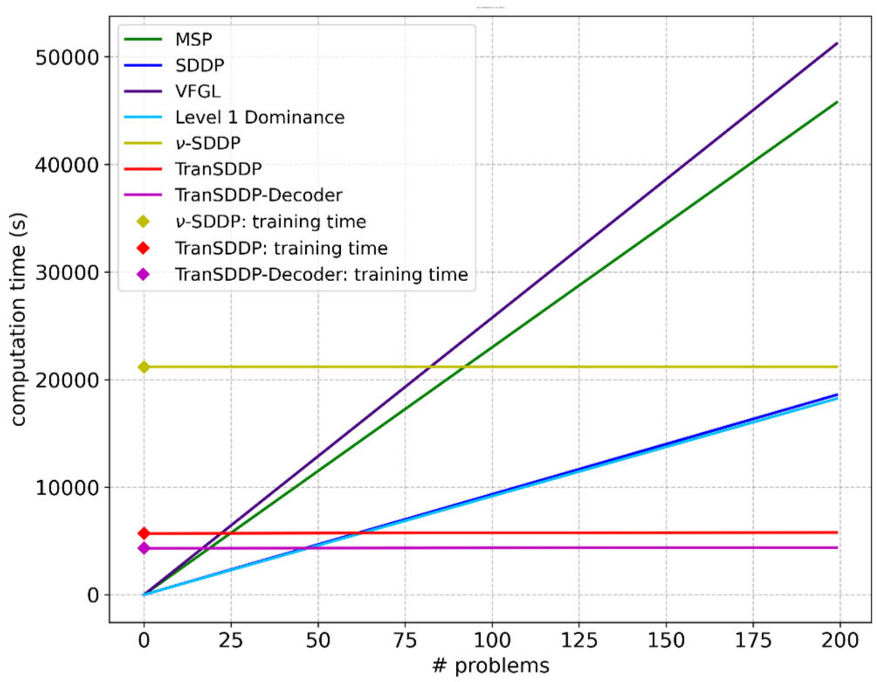


Task	Algorithm	Error ratio
	MSP	-
	SDDP	$1.782 \pm 1.192\%$
	L1	$1.283 \pm 1.096\%$
$T = 7$	VFGL	$0.200 \pm 0.160\%$
	ν-SDDP	$515.722 \pm 0.802\%$
	TranSDDP	$0.962 \pm 0.199\%$
	TranSDDP-Decoder	$0.611 \pm 0.198\%$
	MSP	-
	SDDP	$2.848 \pm 1.647\%$
	L1	$1.630 \pm 1.360\%$
$T = 10$	VFGL	$0.110 \pm 0.082\%$
	ν-SDDP	$317.890 \pm 0.448\%$
	TranSDDP	$1.704 \pm 0.209\%$
	TranSDDP-Decoder	$1.364 \pm 0.208\%$

- Higher error ratio compared to the VFGL
- Outperform other algorithms
- Note that: ν-SDDP has poor performance
  - ν-SDDP can only be applied to linear programs
  - Financial planning problems are not the form of a linear program
- Stable computation time with respect to the number of stages

Experiments: Task 2 – Financial Planning

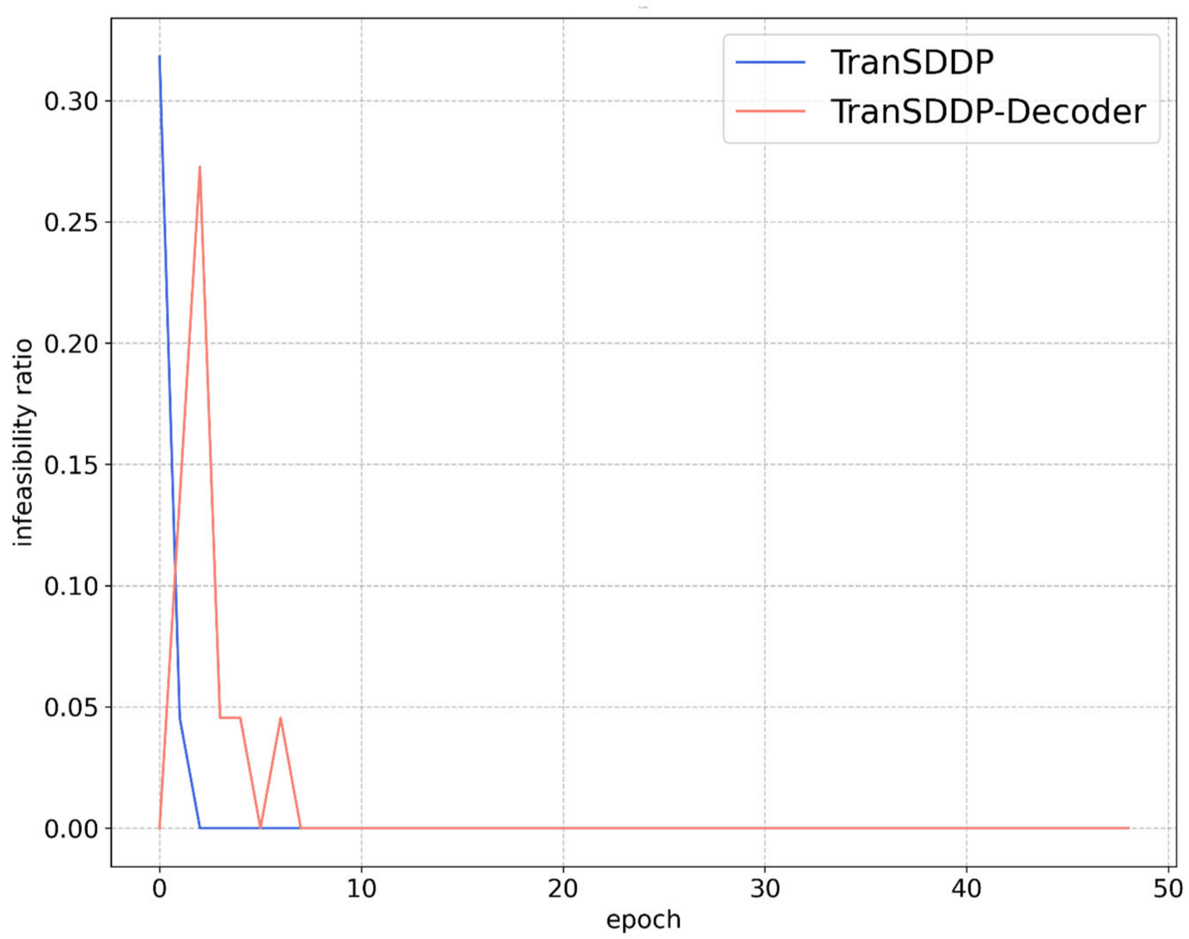
- Computation time: training time + evaluation time



  - The computation time of TranSDDP and TranSDDP-Decoder are almost constant
  - TranSDDP (TranSDDP-Decoder) has a computational advantage over SDDP for 62 (47) or more problems.

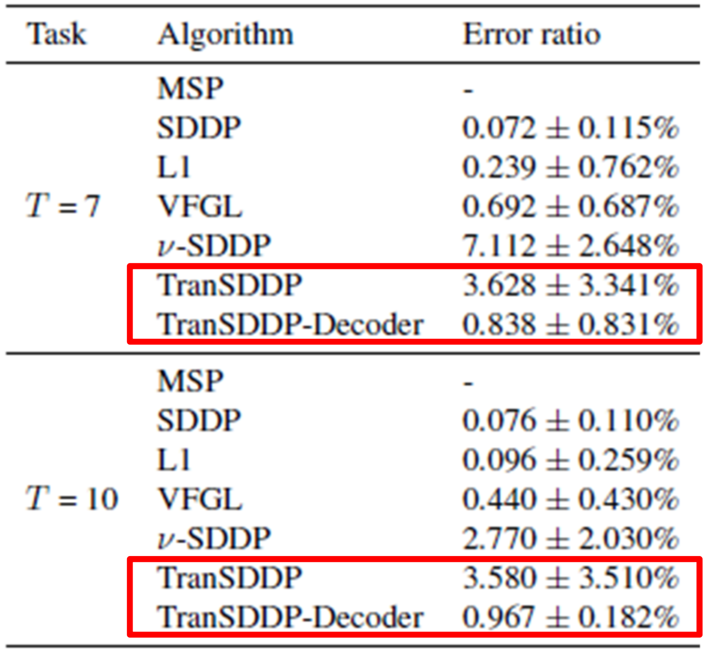
Experiments: Task 2 – Financial Planning

- Infeasibility test



Experiments: Task 3 – Production Planning

- Performance comparison of algorithms for production planning

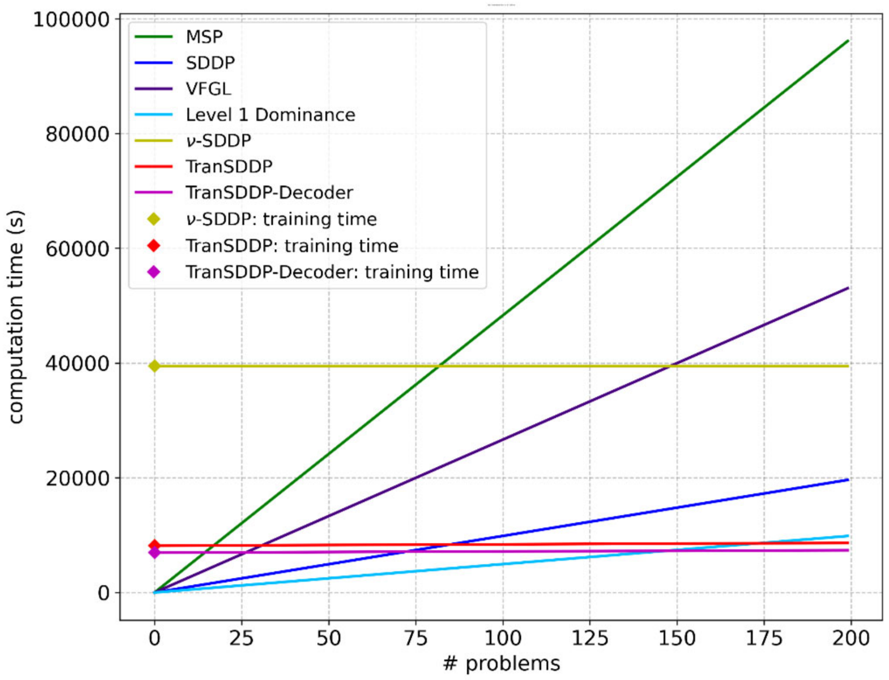


Task	Algorithm	Error ratio
$T = 7$	MSP	-
	SDDP	$0.072 \pm 0.115\%$
	L1	$0.239 \pm 0.762\%$
	VFGL	$0.692 \pm 0.687\%$
	ν-SDDP	$7.112 \pm 2.648\%$
	TranSDDP	$3.628 \pm 3.341\%$
	TranSDDP-Decoder	$0.838 \pm 0.831\%$
$T = 10$	MSP	-
	SDDP	$0.076 \pm 0.110\%$
	L1	$0.096 \pm 0.259\%$
	VFGL	$0.440 \pm 0.430\%$
	ν-SDDP	$2.770 \pm 2.030\%$
	TranSDDP	$3.580 \pm 3.510\%$
	TranSDDP-Decoder	$0.967 \pm 0.182\%$

  - Improve the performance compared to ν-SDDP
  - Higher error ratio than other algorithms
  - Stable computation time with respect to the number of stages

Experiments: Task 3 – Production Planning

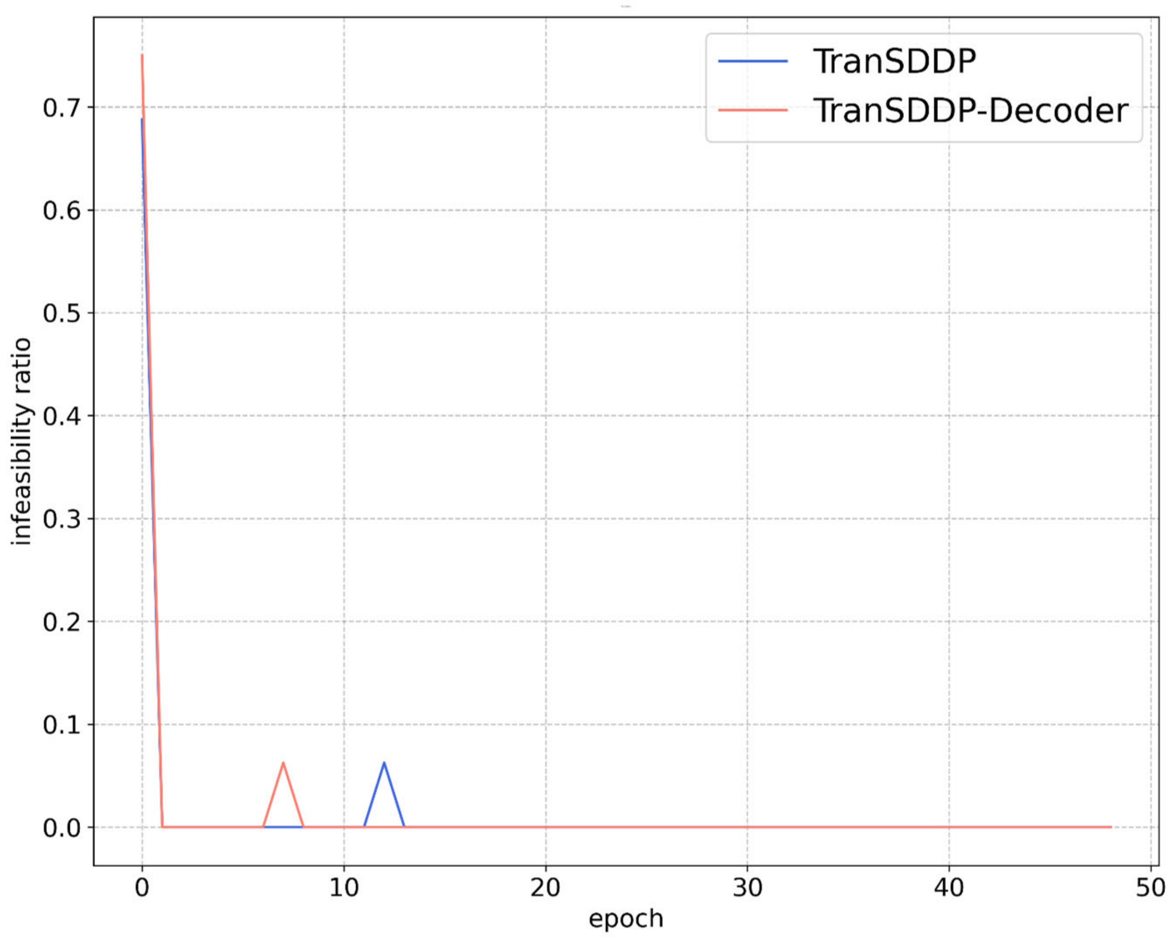
- Computation time: training time + evaluation time



  - The computation time of TranSDDP and TranSDDP-Decoder are almost constant
  - TranSDDP (TranSDDP-Decoder) has a computational advantage over SDDP for 62 (47) or more problems.

Experiments: Task 3 – Production Planning

- Infeasibility test



TranSDDP and TranSDDP-Decoder: Pros and Cons

- The time it takes to solve one problem using our proposed algorithm may be longer compared to the existing algorithms.
- Additionally, the performance of TranSDDP and TranSDDP-Decoder is not significantly better.
- TranSDDP algorithm can provide benefits when there are many problems to be solved and latency is very important.
  - TranSDDP uses the parameters of the probability distribution of the stochastic elements as input, which are sampled from the prior distribution.
  - This allows us to solve all variant problems defined by the parameters that can be sampled from the prior distribution, using the trained model.
  - In other words, we can solve similar problems with slight variations in the distribution of stochastic elements.
- Although training the proposed model takes a long time, the evaluation time, i.e., the time to generate cuts and compute solutions, is very short compared to existing algorithms.
  - Proposed model has an advantage in solving many similar instances of MSOP.
  - Strengths of proposed model in real-world settings is its ability to quickly solve MSOP in situations where latency is tight.
  - This is especially useful in solving problems such as financial planning in cases where the market undergoes sudden changes or energy planning problems in cases where water inflow changes quickly due to flooding, among others

Part 3

DEEP FINANCIAL PLANNING

Based on:

1. Hyunglip Bae, Jang Ho Kim, Woo Chang Kim, On Learning Parametric Optimization with Artificial Neural Networks, *Mathematics of Operations Research*, under revision
2. Hyunglip Bae, Jang Ho Kim, Woo Chang Kim, Deep Financial Planning, Submission ready

A Big Bank in China…

- They had 500 million personal customers (20 times of NPS of Korea).

- They wanted to implement a personalized ALM system on a smartphone app that could generate optimal decision in real time.

  => TranSDDP or TranSDDP-decoder, maybe?

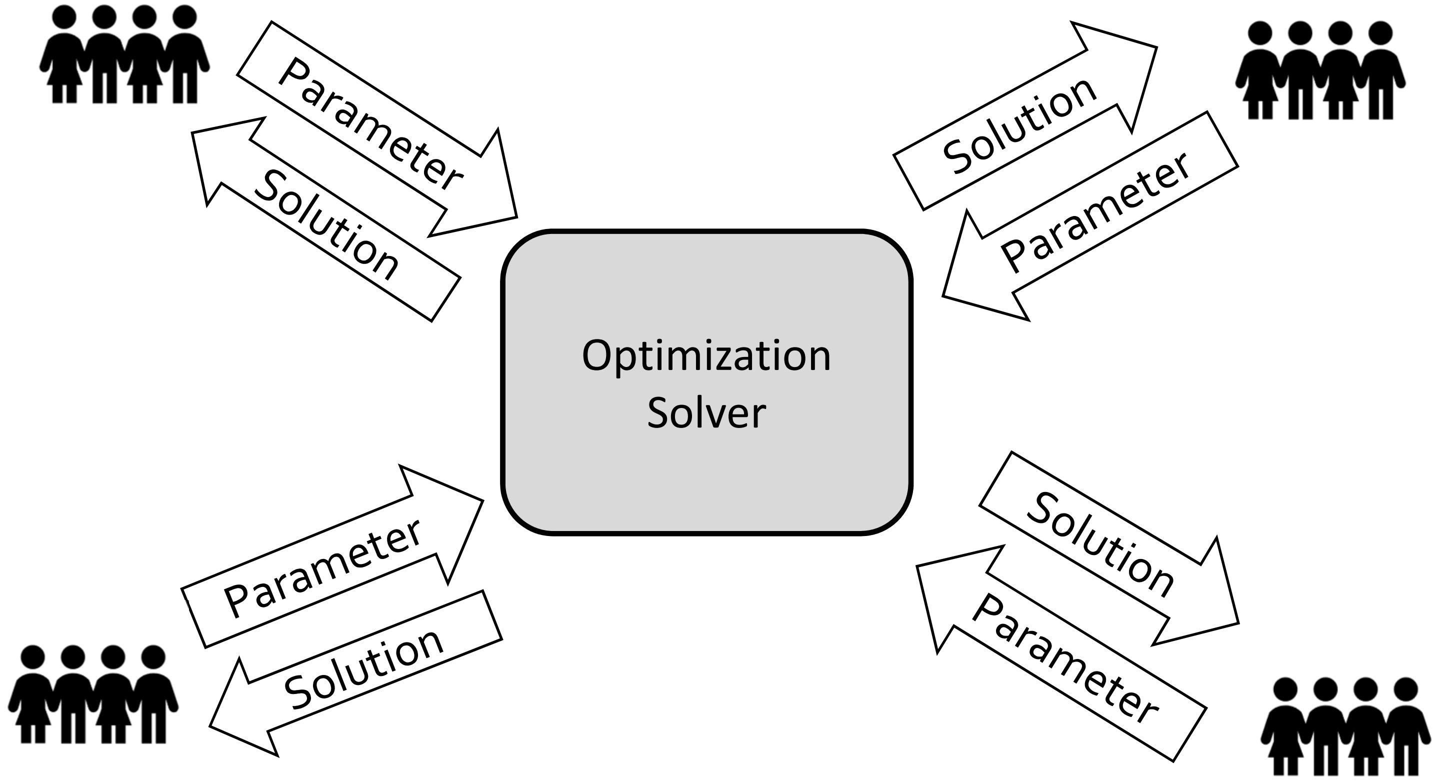
- Also, they liked the idea of goals with priorities.

  => We needed something new

Parametric Optimization: Conventional Approach

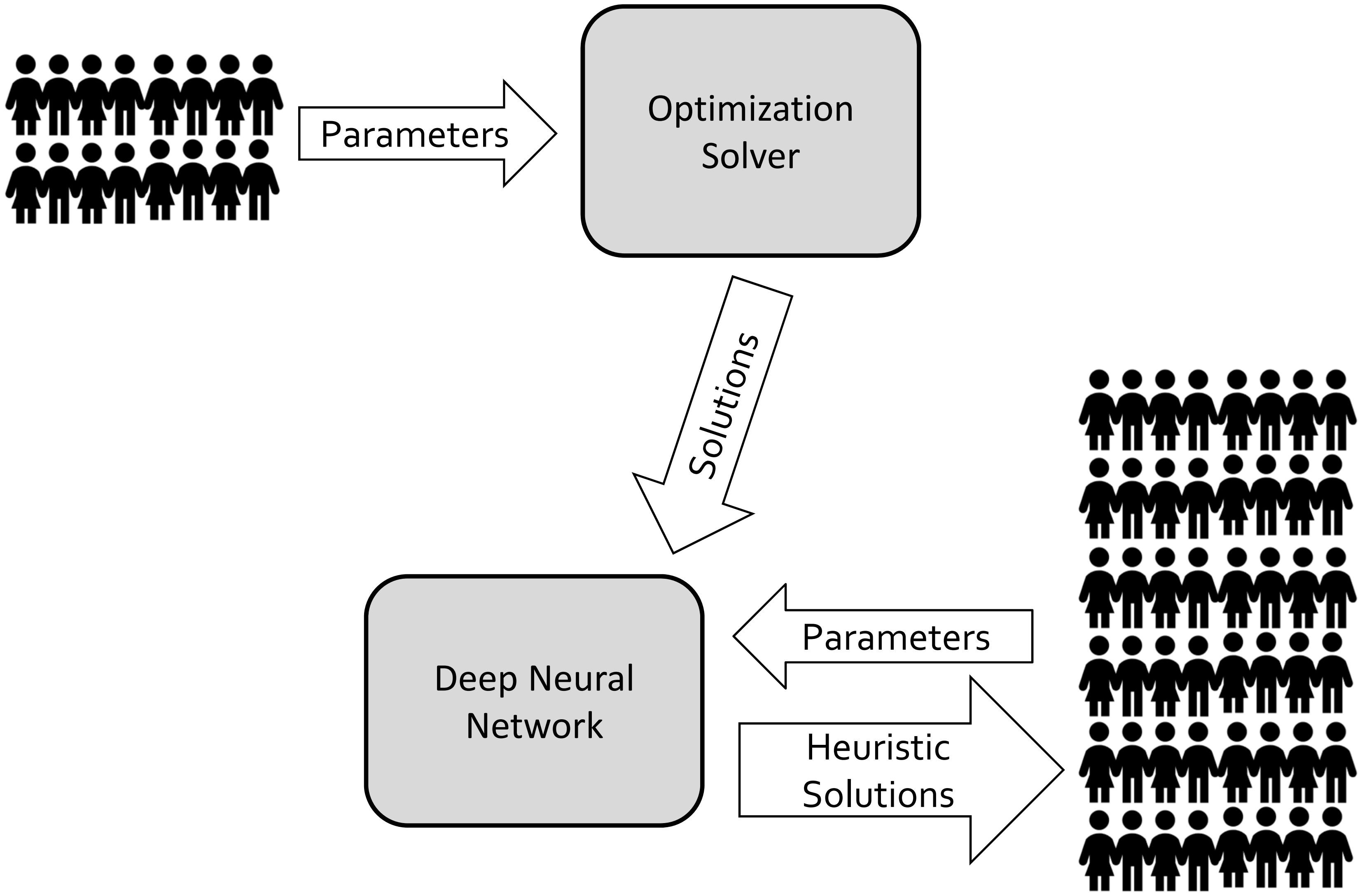
- **Parametric optimization (PO) problem is commonly employed when decisions are made repeatedly as the parameters change**
  - While the problem structure stays the same throughout the entire horizon

- **General framework of PO causes delays between repetitive decisions**



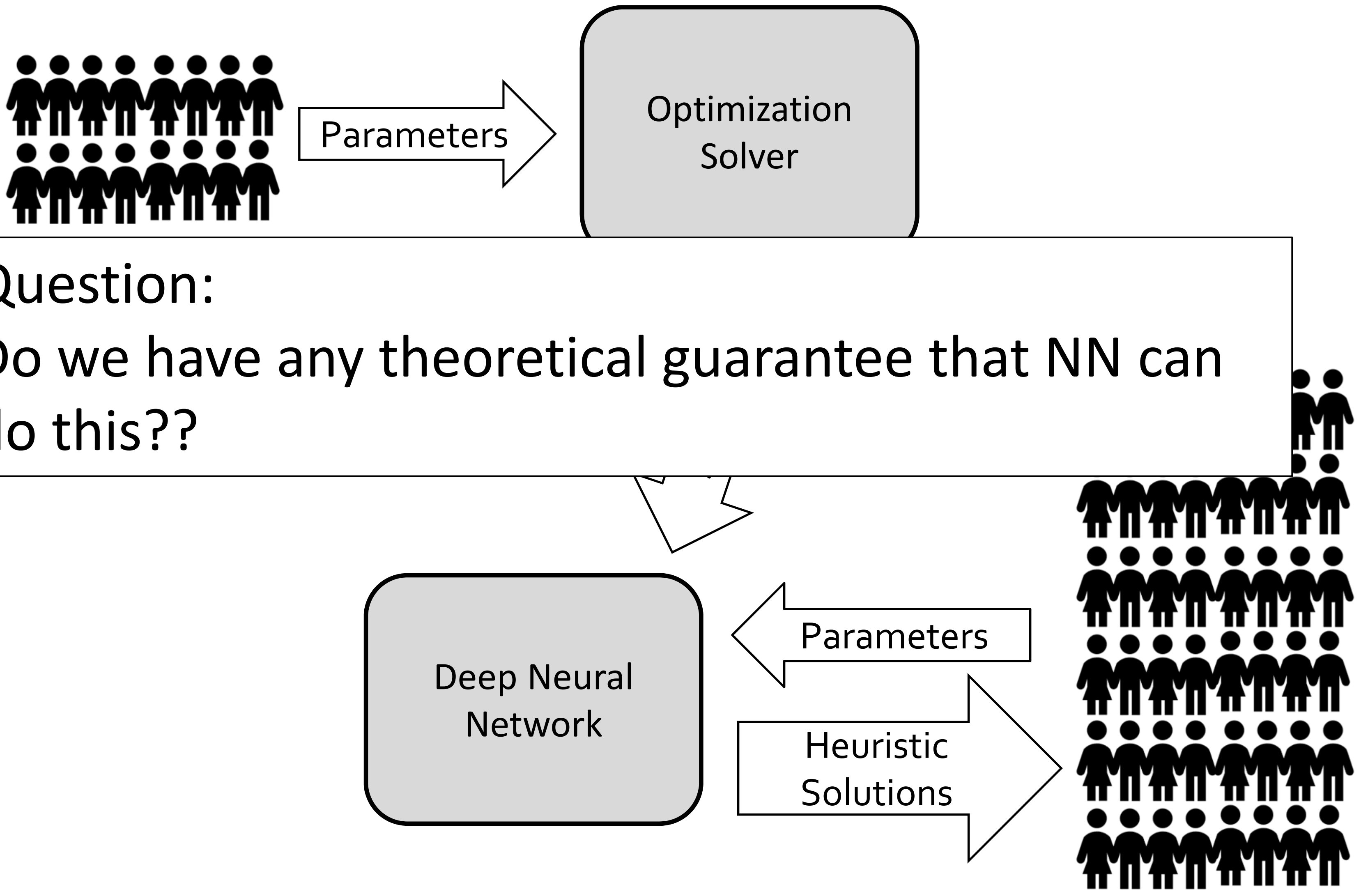
Parametric Optimization: NN-Based Approach

- **Neural Network (NN) can provide heuristics for decision-making**



Parametric Optimization: NN-Based Approach

- **Neural Network (NN) can provide heuristics for decision-making**



Question:
Do we have any theoretical guarantee that NN can do this??

Objective

- **To derive conditions for Universal Approximation Theorem (UAT) to hold for PO problem**
  – Can NN give good approximated solutions for all kinds of PO?

- **Specifically…**
  – We provide sufficient conditions for UAT to hold for value function and optimal policy for continuous PO problems

  – We also address situations when these sufficient conditions are not satisfied. In particular, we define a sampling function and its stability which makes good approximation possible through NN even without the sufficient conditions in original problems.

  – We directly link vast amount of literature on NN with approximating optimization problems.

Parametric Optimization

- **Parametric optimization takes the form**

$$\begin{array}{ll} maximize & f(x|\theta) \\ subject\ to & x \in C(\theta) \end{array}$$

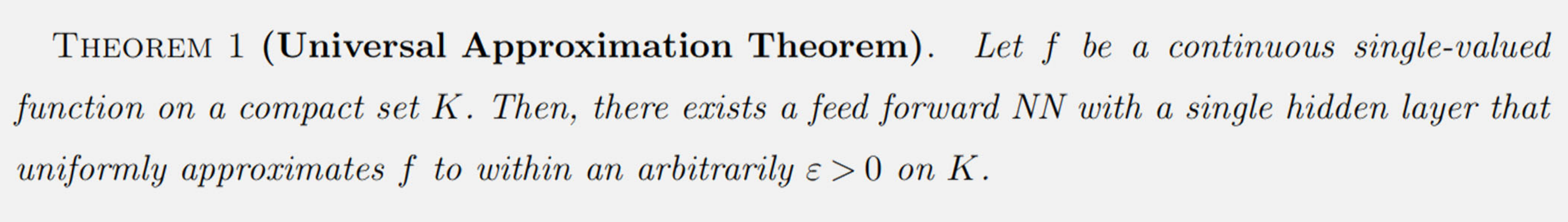
  - $x \in X \subset \mathbb{R}^n$ is the decision variable, $\theta \in \Theta \subset \mathbb{R}^k$ is the parameter
  - $f: \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}$ is the objective function and $C: \mathbb{R}^k \rightrightarrows 2^{\mathbb{R}^n}$ is a multivalued mapping, or correspondence, representing the feasible region

- **Optimal value function**
  - $f^*(\theta) = \max\{f(x|\theta): x \in C(\theta)\}$

- **Optimal policy correspondence (multi-valued function)**
  - $C^*(\theta) = arg \max\{f(x|\theta): x \in C(\theta)\}$
  - An optimal solution $x^*(\theta)$ is an element of $C^*(\theta)$

Universal Approximation Theorem

- **Universal Approximation Theorem (UAT)**
  - Result about capability of Neural Network as approximator
  - While there are many variations, the point is that function expressed as Neural Network is dense on the function space of interest
  - We use the most classical versions of UAT by Hornik et al., Cybenko and Funahashi.



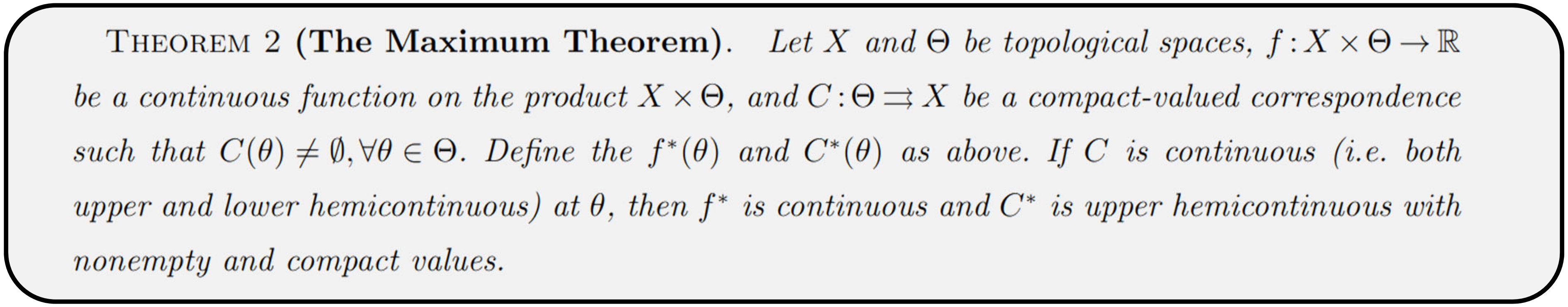
THEOREM 1 **(Universal Approximation Theorem).** *Let f be a continuous single-valued function on a compact set K. Then, there exists a feed forward NN with a single hidden layer that uniformly approximates f to within an arbitrarily $\varepsilon > 0$ on K.*

- **Note that UAT requires function such that**
  - 1) Continuous
  - 2) Single-valued

- **Thus, we aim to find "continuous single-valued function" for PO approximation**

The Maximum Theorem

- **The Maximum Theorem (Berge, 1963) state that if**
  - 1) $C(\theta)$ is compact set and non-empty
  - 2) f is continuous
  - 3) $C(\theta)$ is continuous
  - Then, f^* is continuous and C^* is upper-hemi continuous



THEOREM 2 **(The Maximum Theorem).** *Let X and Θ be topological spaces, $f: X \times \Theta \to \mathbb{R}$ be a continuous function on the product $X \times \Theta$, and $C: \Theta \rightrightarrows X$ be a compact-valued correspondence such that $C(\theta) \neq \emptyset, \forall \theta \in \Theta$. Define the $f^*(\theta)$ and $C^*(\theta)$ as above. If C is continuous (i.e. both upper and lower hemicontinuous) at θ, then f^* is continuous and C^* is upper hemicontinuous with nonempty and compact values.*

- **Thus, f^* satisfies conditions for UAT**

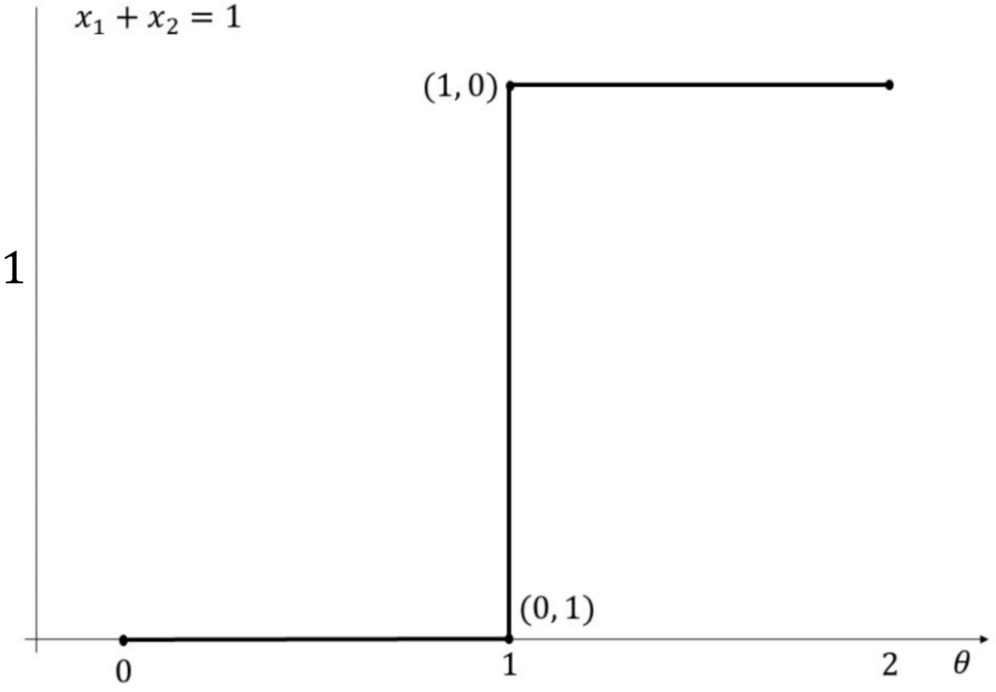
- **Still, C^* is not guaranteed to be continuous and is not even single-valued function**
  - If C^* is singleton for each θ, NN does the job.

Continuous selection of C^*

- **However, some important classes of optimization, including linear programming problems, do not necessarily have lower hemi-continuous optimal policy**
  - Ex)

$$\begin{array}{ll} \textit{minimize} & -\theta x_1 - x_2 \\ \textit{subject to} & x_1 + x_2 \le 1, x_1 \ge 0, x_2 \ge 0 \end{array}$$

$\boldsymbol{C}^*(\theta)$ on $x_1 + x_2 = 1$



How to Sample $\boldsymbol{x^*(\theta_i)}$ from $\boldsymbol{C^*(\theta_i)}$?

- A training example for optimal policy is a pair of a parameter and its corresponding optimal solution $\left(\theta_i, x^*(\theta_i)\right)$.

- Let the training data be a set of examples, $T = \left\{\left(\theta_i, x^*(\theta_i)\right) \middle| i = 1, \cdots, m\right\}$.

- Notice that there can be more than one optimal solution for each θ_i.
  - In practice, it is computationally expensive, if not impossible, to obtain the entire set of optimal solutions.
  - In fact, it is difficult even to identify whether there are multiple optimal solutions or not.

- We assume that there exists a solver that can extract exactly one element $x^*(\theta_i)$ from $C^*(\theta_i)$ for any give θ_i.

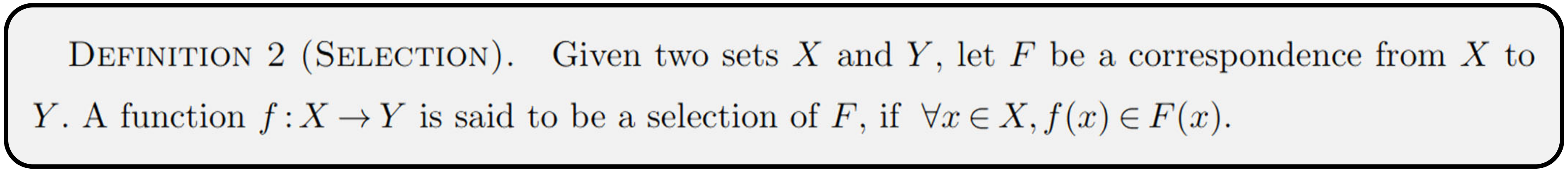
- However, it does not have control on the choice of $x^*(\theta_i)$, so that the optimal solution is obtained in a random manner from $C^*(\theta_i)$.

- Moreover, the solver is not able to identify if $C^*(\theta_i)$ is a singleton or not.

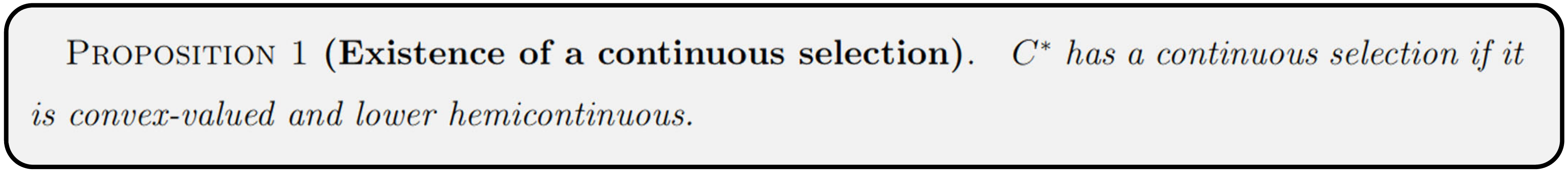
- It is as if the training data is a discrete sample path from the correspondence C^* indexed by $\{\theta_i | i = 1, \cdots, m\}$.

Continuous selection of C^*

- **When can we find continuous single-valued function in C^*?**
  - Under some conditions on C^*, it is possible to find a continuous function called a “selection”



DEFINITION 2 (SELECTION). Given two sets X and Y, let F be a correspondence from X to Y. A function $f: X \to Y$ is said to be a selection of F, if $\forall x \in X, f(x) \in F(x)$.



PROPOSITION 1 **(Existence of a continuous selection)**. *C^* has a continuous selection if it is convex-valued and lower hemicontinuous.*

Construction of continuous function for C^*

- **One idea is to construct continuous single-valued function for $\boldsymbol{C^*}$**
  - Let $T = \{(\theta_i, x^*(\theta_i)) | i = 1, \cdots, m\}$ be training data
  - Denote convex hull of $\theta_1, \theta_2, \dots, \theta_m$ as $Conv(\{\theta_1, \theta_2, \dots, \theta_m\})$
  - We consider a finite collection $S = \{S_1, \dots, S_d\}$ of subset of $\{\theta_1, \theta_2, \dots, \theta_m\}$ s.t

a) For each $i \in \{1, \dots, d\}$, $|S_i| = k + 1$ i.e. $Conv(S_i)$ is k-dimensional simplex
b) $Conv(\{\theta_1, \theta_2, \dots, \theta_m\}) = \bigcup_{i=1}^{d} Conv(S_i)$
c) $Conv(S_i) \cap Conv(S_j) = Conv(S_i \cap S_j)$

- **Given such collection $\boldsymbol{S}$, $\boldsymbol{\theta} \in \boldsymbol{Conv}(\{\boldsymbol{\theta_1}, \boldsymbol{\theta_2}, \dots, \boldsymbol{\theta_m}\})$ can be expressed as**

$$\theta = \lambda_1^i \theta_1^i + \cdots + \lambda_{k+1}^i \theta_{k+1}^i, \{\theta_1^i, \dots, \theta_{k+1}^i\} = S_i \in S$$
$$\lambda_1^i, \dots, \lambda_{k+1}^i \geq 0, \lambda_1^i + \cdots + \lambda_{k+1}^i = 1$$

  - Our key idea is approximate $x^*(\theta)$ as $\lambda_1^i x^*(\theta_1^i) + \cdots + \lambda_{k+1}^i x^*(\theta_{k+1}^i)$

Main Result: sufficient conditions for convergence

- **Definition of ε-suboptimality and ε-infeasibility**

DEFINITION 3. Let f be the function and C be the correspondence in the formulation given by (1). Then,

(a) $\hat{x}(\theta)$ is ε-infeasible solution if $\hat{x}(\theta) \in \mathcal{B}_\varepsilon(C(\theta))$

(b) $\hat{x}(\theta)$ is ε-suboptimal solution if $|f(\hat{x}(\theta),\theta) - f(x^*(\theta),\theta)| < \varepsilon$.

- **Main Theorem**

THEOREM 3. *Suppose that f, C satisfy all conditions for Theorem 2. Define a training data set $T = \{(\theta_i, x^*(\theta_i)) | \theta_i \in \Theta, i = 1, \cdots, m\}$ where $x^*(\theta_i)$ is an arbitrarily chosen point from $C^*(\theta_i)$. For $\theta \in Conv(\{\theta_1, \theta_2, \ldots, \theta_m\})$, consider a finite collection S as above and its one element $S_j = \{\theta_1^j, \ldots, \theta_{k+1}^j\}$. For $\theta \in Conv(S_j)$, write $\theta = \lambda_1^j \theta_1^j + \ldots + \lambda_{k+1}^j \theta_{k+1}^j$. Define $diam(S_j) = \max\{\|p - q\| : p, q \in Conv(S_j)\}$ and $\hat{x}(\theta) = \lambda_1^j x^*(\theta_1^j) + \ldots + \lambda_{k+1}^j x^*(\theta_{k+1}^j)$. Then, the followings hold.*

(a) If $Conv(C^*(\theta)) \subseteq C(\theta)$, for given $\varepsilon > 0$, there exists $\delta > 0$ such that if $diam(S_j) < \delta$, $\hat{x}(\theta)$ is ε-infeasible solution.

(b) If $f(x,\theta) = f^*(\theta), \forall x \in Conv(C^*(\theta))$, for given $\varepsilon > 0$, there exists $\delta > 0$ such that if $diam(S_j) < \delta$, $\hat{x}(\theta)$ is ε-suboptimal solution

Proof for (a) of Theorem 3

Proof (a) Assume $Conv(C^*(\theta)) \subseteq C(\theta)$ and let $\varepsilon > 0$. Since f, C satisfy all conditions for Theorem 2, C^* is upper hemicontinuous. Thus, a set

$$\Delta_\theta = \{\delta > 0 : \|\theta - \theta'\| < \delta \Longrightarrow C^*(\theta') \subseteq \mathcal{B}_\varepsilon(C^*(\theta))\}$$

is not empty.

Define $\delta_\theta = \sup \Delta_\theta$. Choose $\delta = \delta_\theta$. If $diam(S_j) < \delta$, $\|\theta - \theta_l^j\| < \delta$ i.e. $C^*(\theta_l^j) \subseteq \mathcal{B}_\varepsilon(C^*(\theta)), \forall l = 1, \ldots, k+1$. Then, with the assumption and $C^*(\theta) \subseteq Conv(C^*(\theta))$,

$$x^*(\theta_l^j) \in C^*(\theta_j') \subseteq \mathcal{B}_\varepsilon(C^*(\theta)) \subseteq \mathcal{B}_\varepsilon(Conv(C^*(\theta))) \subseteq \mathcal{B}_\varepsilon(C(\theta)), \forall l = 1, 2, \ldots, k+1$$

Thus, $x^*(\theta_l^j) \in \mathcal{B}_\varepsilon(Conv(C^*(\theta))) \subseteq \mathcal{B}_\varepsilon(C(\theta))$. Note that, since $Conv(C^*(\theta))$ is convex set, $\mathcal{B}_\varepsilon(Conv(C^*(\theta)))$ is also convex set. This means the convex combination

$$\lambda_1^j x^*(\theta_1^j) + \ldots + \lambda_{k+1}^j x^*(\theta_{k+1}^j)$$

is in $\mathcal{B}_\varepsilon(Conv(C^*(\theta)))$. Thus, $\hat{x}(\theta) \in B_\varepsilon(C(\theta))$.

Proof for (b) of Theorem 3

(b) Assume $f(x,\theta) = f^*(\theta), \forall x \in Conv(C^*(\theta))$ and let $\varepsilon > 0$. We first show there exists $\delta' > 0$ such that,

$$\inf_{x^* \in Conv(C^*(\theta))} \|x' - x^*\| < \delta' \Longrightarrow |f(x',\theta) - f^*(\theta)| < \varepsilon, \forall x' \in \mathcal{B}_\varepsilon(C(\theta)).$$

Since C is compact-valued correspondence, $\mathcal{B}_\varepsilon(C(\theta)) \times \theta$ is compact set. Thus, f is uniformly continuous on $\mathcal{B}_\varepsilon(C(\theta)) \times \theta$. Thus, there exist $\delta'' > 0$ such that,

$$\|y - z\| < \delta'' \Longrightarrow |f(y,\theta) - f(z,\theta)| < \varepsilon, \forall y, z \in \mathcal{B}_\varepsilon(C(\theta))$$

Choose $\delta' = \delta''$. Note that C^* is compact-valued correspondence from Theorem 2. Thus, $Conv(C^*(\theta))$ is also compact set from Lemma 1. Hence,

$$x^*_{min} = \arg\min_{x^* \in Conv(C^*(\theta))} \|x' - x^*\| = \arg\min_{x^* \in Conv(C^*(\theta))} \|x' - x^*\|$$

is in $Conv(C^*(\theta))$. Now we have

$$\inf_{x^* \in Conv(C^*(\theta))} \|x' - x^*\| < \delta' \Longrightarrow \|x' - x^*_{min}\| < \delta' = \delta'' \Longrightarrow |f(x',\theta) - f(x^*_{min},\theta)| < \varepsilon.$$

Since $x^*_{min} \in Conv(C^*(\theta))$, $f(x^*_{min},\theta) = f^*(\theta)$. Thus, $|f(x',\theta) - f^*(\theta)| < \varepsilon$

Proof for (b) of Theorem 3

Now, we prove part (b) of the theorem. From the above statement, there exists $\delta' > 0$ such that,

$$\inf_{x^* \in Conv(C^*(\theta))} \|x' - x^*\| < \delta' \Longrightarrow |f(x',\theta) - f^*(\theta)| < \varepsilon, \forall x' \in \mathcal{B}_\varepsilon(C(\theta))$$

Also, since f, C satisfy all conditions for Theorem 2, C^* is upper hemicontinuous. Thus, a set

$$\Delta_\theta = \{\delta > 0 | \, if \; \|\theta - \theta'\| < \delta, C^*(\theta') \subseteq \mathcal{B}'_\delta(C^*(\theta))\}$$

is not empty.

Define $\delta_\theta = \sup \Delta_\theta$. Choose $\delta = \delta_\theta$. If $diam(S_j) < \delta$, $\|\theta - \theta_l^j\| < \delta$ i.e. $C^*(\theta_l^j) \subseteq \mathcal{B}_{\delta'}(C^*(\theta)), \forall l = 1, \ldots, k+1$. Then with $C^*(\theta) \subseteq Conv(C^*(\theta))$,

$$x^*(\theta_l^j) \in C^*(\theta_l^j) \subseteq \mathcal{B}_{\delta'}(C^*(\theta)) \subseteq \mathcal{B}_{\delta'}(Conv(C^*(\theta))), \forall l = 1, \ldots, k+1$$

Since $Conv(C^*(\theta))$ is convex set, $\mathcal{B}_{\delta'}(Conv(C^*(\theta)))$ is also convex set. This means the convex combination

$$\hat{x}(\theta) = \lambda_1^j x^*(\theta_1^j) + \ldots + \lambda_{k+1}^j x^*(\theta_{k+1}^j)$$

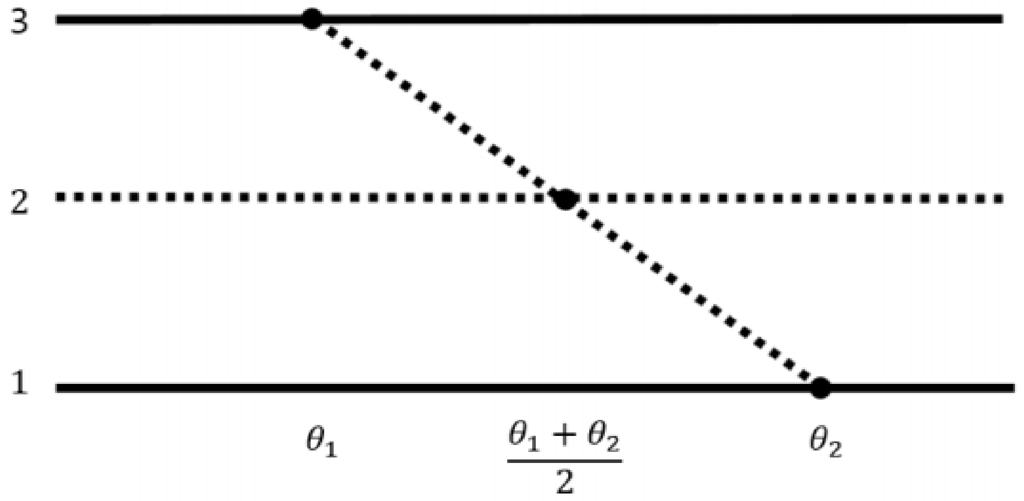
is in $\mathcal{B}_{\delta'}(Conv(C^*(\theta)))$. Also, note that $\hat{x}(\theta) \in \mathcal{B}_\varepsilon(C(\theta))$ from part (a) since assumption in (b) indicates $Conv(C^*(\theta)) \subseteq C^*(\theta) \subseteq C(\theta)$. Accordingly, $|f(\hat{x}(\theta),\theta) - f^*(\theta)| < \varepsilon$. $\square$

Construction of continuous function for C^*

- **Since $\boldsymbol{x}^*(\boldsymbol{\theta})$ is arbitrarily chosen from $\boldsymbol{C}^*(\boldsymbol{\theta})$, there exist PO problems where convergence of errors is not guaranteed**

$$\begin{aligned} minimize \quad & f(x,\theta) = (x-1)^2(x-3)^2 \\ subject\ to \quad & C(\theta) = [1,3] \end{aligned}$$

  - Then $C^*(\theta) = \{1,3\},\ \forall\theta$



  - The construction always has suboptimality 1 $(f^*((\theta_1+\theta_2)/2) - f(2,(\theta_1+\theta_2)/2) = 1)$

Stable Sampling Function

- **Definition of stable sampling function**

DEFINITION 4. Define a sampling function $s:\Theta \to \bigcup_\theta C^*(\theta)$ as $s(\theta) = x^*(\theta)$. Sampling function s is stable with respect to C^* if there exists a non-empty, compact, and convex-valued upper hemicontinuous correspondence $\overline{C^*}$ such that $s(\theta) \in \overline{C^*}(\theta) \subseteq C^*(\theta)\ \forall\theta$.

- **If stable sampling function exists, the sufficient conditions of main theorem become redundant**

THEOREM 4. *Let s be a stable sampling function and $\overline{C^*}$ be the non-empty, compact, and convex-valued upper hemicontinuous correspondence with respect to C^*. Then the followings hold.*

(a) $Conv(\overline{C^*}(\theta)) \subseteq C(\theta)$

(b) $f(x,\theta) = f^*(\theta), \forall x \in Conv(\overline{C^*}(\theta))$

Proof for Theorem 4

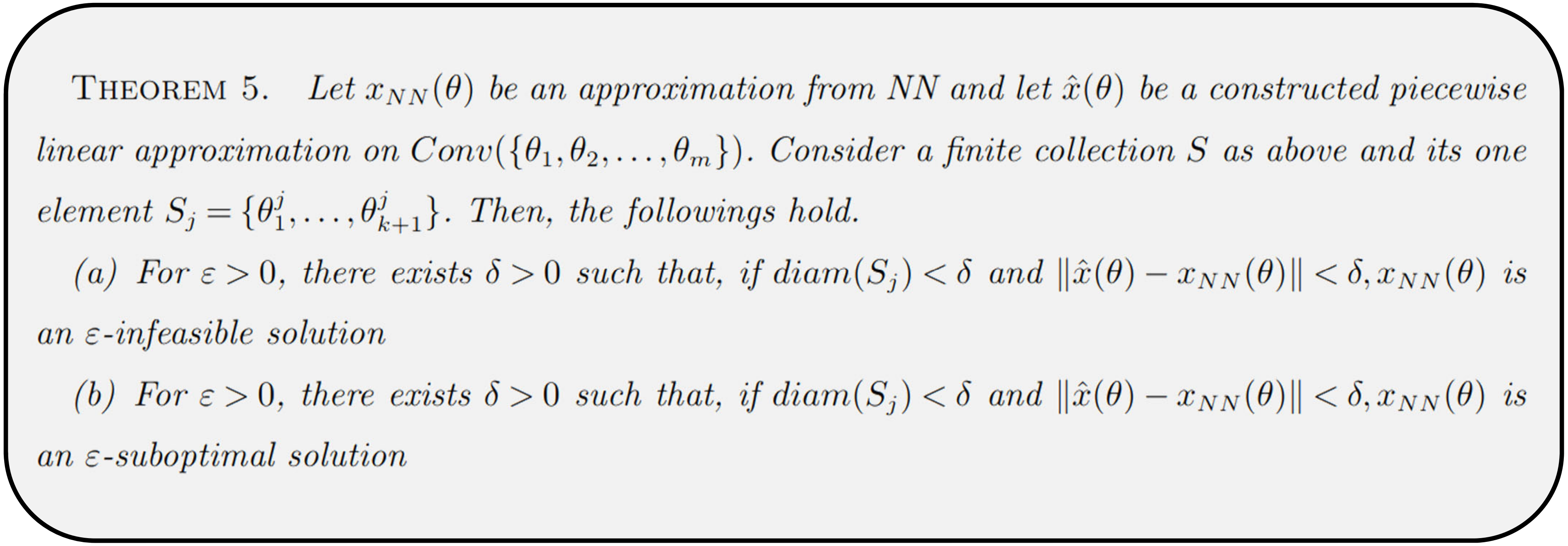
Proof Since $\overline{C^*}(\theta)$ is convex, $Conv(\overline{C^*}(\theta)) = \overline{C^*}(\theta)$. With this fact, we have

(a) $Conv(\overline{C^*}(\theta)) = \overline{C^*}(\theta) \subseteq C^*(\theta) \subseteq C(\theta)$.

(b) $\overline{C^*}(\theta) \subseteq C^*(\theta)$ implies $f(x,\theta) = f^*(\theta), \forall x \in Conv(\overline{C^*}(\theta))$. $\square$

Approximation of Target Function with NN

- **We approximate constructed continuous single-valued function with NN**
  - **If NN is close to target function sufficiently, this is good approximation for PO**



THEOREM 5. *Let $x_{NN}(\theta)$ be an approximation from NN and let $\hat{x}(\theta)$ be a constructed piecewise linear approximation on $Conv(\{\theta_1, \theta_2, \ldots, \theta_m\})$. Consider a finite collection S as above and its one element $S_j = \{\theta_1^j, \ldots, \theta_{k+1}^j\}$. Then, the followings hold.*

(a) For $\varepsilon > 0$, there exists $\delta > 0$ such that, if $diam(S_j) < \delta$ and $\|\hat{x}(\theta) - x_{NN}(\theta)\| < \delta, x_{NN}(\theta)$ is an ε-infeasible solution

(b) For $\varepsilon > 0$, there exists $\delta > 0$ such that, if $diam(S_j) < \delta$ and $\|\hat{x}(\theta) - x_{NN}(\theta)\| < \delta, x_{NN}(\theta)$ is an ε-suboptimal solution

Proof for (a) of Theorem 5

Proof (a) From part (a) of Theorem 3, for $\varepsilon/2 > 0$, there exists $\delta_1 > 0$ such that,

$$diam(S_j) < \delta_1 \Longrightarrow \hat{x}(\theta) \in \mathcal{B}_{\varepsilon/2}(C(\theta))$$

Choose $\delta = \min\{\delta_1, \varepsilon/2\}$. Then, we have

$$diam(S_j) < \delta, \|\hat{x}(\theta) - x_{NN}(\theta)\| < \delta \Longrightarrow \hat{x}(\theta) \in \mathcal{B}_{\varepsilon/2}(C(\theta)), x_{NN}(\theta) \in \mathcal{B}_{\varepsilon/2}(C(\theta))$$

Thus, $x_{NN}(\theta) \in \mathcal{B}_{\varepsilon/2}(\hat{x}(\theta)) \subseteq \mathcal{B}_{\varepsilon}(C(\theta))$

Proof for (b) of Theorem 5

(b) From part (b) of Theorem 3, for $\varepsilon/2 > 0$, there exists $\delta_1 > 0$ such that,

$$diam(S_j) < \delta_1 \Longrightarrow |f(\hat{x}(\theta), \theta) - f(x^*(\theta), \theta)| < \varepsilon/2$$

Also, since f is continuous, there exists $\delta_2 > 0$ such that,

$$\|\hat{x}(\theta) - x_{NN}(\theta)\| < \delta_2 \Longrightarrow |f(\hat{x}(\theta), \theta) - f(x_{NN}(\theta), \theta)| < \varepsilon/2$$

Choose $\delta = \min\{\delta_1, \delta_2\}$. Then, if $diam(S_j) < \delta$ and $\|\hat{x}(\theta) - x_{NN}(\theta)\| < \delta$,

$$|f(x^*(\theta), \theta) - f(x_{NN}(\theta), \theta)|$$

$$= |f(x^*(\theta), \theta) - f(\hat{x}(\theta), \theta) + f(\hat{x}(\theta), \theta) - f(x_{NN}(\theta), \theta)|$$

$$\leq |f(x^*(\theta), \theta) - f(\hat{x}(\theta), \theta)| + |f(\hat{x}(\theta), \theta) - f(x_{NN}(\theta), \theta)|$$

$$< \varepsilon/2 + \varepsilon/2 = \varepsilon$$

Three financial planning models: Simple ALM

- **Simple ALM problem**
  - Investor makes an investment at the beginning to achieve a single goal at the final stage.

Parameters for investor:

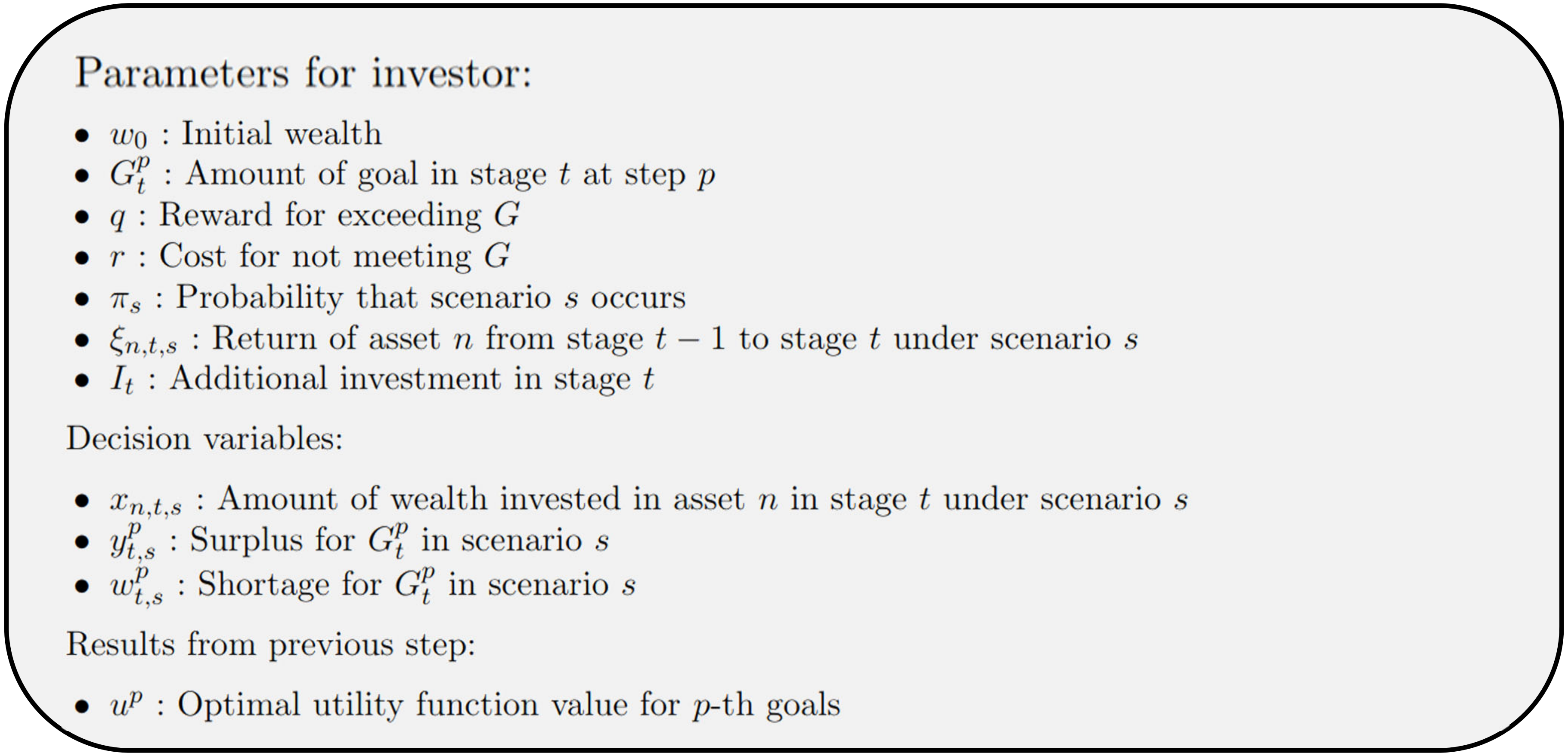
- w_0 : Initial wealth
- G : Amount of goal
- q : Reward for exceeding G
- r : Cost for not meeting G
- π_s : Probability that scenario s occurs
- $\xi_{n,t,s}$: Return of asset n from stage $t-1$ to stage t under scenario s

Decision variables:

- $x_{n,t,s}$: Amount of wealth invested in asset n in stage t under scenario s
- y_s : Surplus for G in scenario s
- w_s : Shortage for G in scenario s

Three financial planning models: Simple GBI

- **Simple GBI problem**
  - Slight extension of the Simple ALM problem by adding additional goals with different priority levels and additional investment I_t in stage t
  - We assume that only one goal can be placed in each stage



Parameters for investor:

- w_0 : Initial wealth
- G_t^p : Amount of goal in stage t at step p
- q : Reward for exceeding G
- r : Cost for not meeting G
- π_s : Probability that scenario s occurs
- $\xi_{n,t,s}$: Return of asset n from stage $t-1$ to stage t under scenario s
- I_t : Additional investment in stage t

Decision variables:

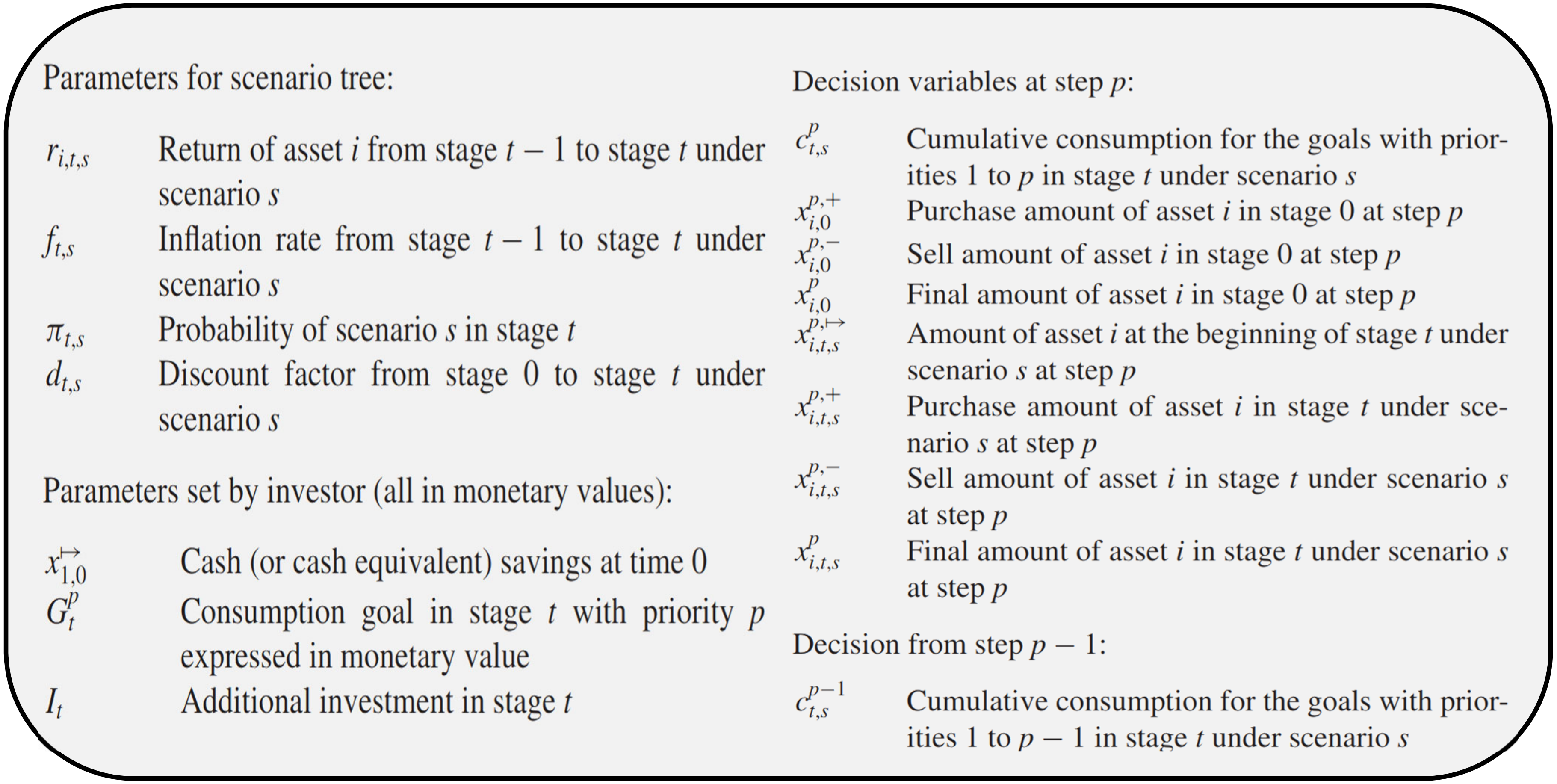
- $x_{n,t,s}$: Amount of wealth invested in asset n in stage t under scenario s
- $y_{t,s}^p$: Surplus for G_t^p in scenario s
- $w_{t,s}^p$: Shortage for G_t^p in scenario s

Results from previous step:

- u^p : Optimal utility function value for p-th goals

Three financial planning models: Advanced GBI

- **Advanced GBI problem**
  - Model proposed by "Personalized goal-based investing via multi-stage stochastic goal programming (Kim et al., 2020)"



Parameters for scenario tree:

$r_{i,t,s}$	Return of asset i from stage $t-1$ to stage t under scenario s
$f_{t,s}$	Inflation rate from stage $t-1$ to stage t under scenario s
$\pi_{t,s}$	Probability of scenario s in stage t
$d_{t,s}$	Discount factor from stage 0 to stage t under scenario s

Parameters set by investor (all in monetary values):

$x_{1,0}^{\mapsto}$	Cash (or cash equivalent) savings at time 0
G_t^p	Consumption goal in stage t with priority p expressed in monetary value
I_t	Additional investment in stage t

Decision variables at step p:

$c_{t,s}^p$	Cumulative consumption for the goals with priorities 1 to p in stage t under scenario s
$x_{i,0}^{p,+}$	Purchase amount of asset i in stage 0 at step p
$x_{i,0}^{p,-}$	Sell amount of asset i in stage 0 at step p
$x_{i,0}^{p}$	Final amount of asset i in stage 0 at step p
$x_{i,t,s}^{p,\mapsto}$	Amount of asset i at the beginning of stage t under scenario s at step p
$x_{i,t,s}^{p,+}$	Purchase amount of asset i in stage t under scenario s at step p
$x_{i,t,s}^{p,-}$	Sell amount of asset i in stage t under scenario s at step p
$x_{i,t,s}^{p}$	Final amount of asset i in stage t under scenario s at step p

Decision from step $p-1$:

$c_{t,s}^{p-1}$	Cumulative consumption for the goals with priorities 1 to $p-1$ in stage t under scenario s

Target of Neural Networks

- **We aim to approximate scenario-wise expectation of**
  - Asset allocation weights $E_s[a]$
  - Goal allocation weight $E_s[g]$
  - Goal achievement rate for each stage $E_s[c]$
  - n: asset, t: stage, s: scenario, p: priority, σ_p: stage corresponding priority p

- **Simple ALM**

$$f(G, w_0) = \left(E_s\left[a_{n,t,s}\right], E_s[g_s], E_s[c_s]\right),$$
$$\forall n = 1, \dots, N, \forall t = 1, \dots, T$$

- **Simple and Advanced GBI**

$$f\left(G_t^p, w_0, I_t\right) = \left(E_s\left[a_{n,t,s}\right], E_s\left[g_{\sigma_p,s}^p\right], E_s\left[c_{\sigma_p,s}^p\right]\right),$$
$$\forall n = 1, \dots, N, \forall t = 1, \dots, T$$

Theoretical foundation for Deep Financial Planning

- **Parametric optimization takes the form**

$$\begin{array}{ll} maximize & f(x,\theta) \\ subject\ to & x \in F(\theta) \end{array}$$

- **From “On Learning Parametric Optimization with Artificial Neural Networks”, for good approximation, we need**
  - Upper hemi-continuity of $F^*(\theta)$
  - For upper hemi-continuity of $F^*(\theta)$, we need continuity of $F(\theta)$

- **Suppose there are total $\boldsymbol{n}$ goals. Let $\boldsymbol{f_i(x,\theta)}$ be objective function for $\boldsymbol{i}$-th goal and $\boldsymbol{F(\theta)}$ be initial feasible region.**
  - Then, goal programming proceeds by sequentially solving the following optimization problem from $i = 1$ to n

$$\begin{array}{ll} maximize & f_i(x,\theta) \\ subject\ to & x \in F_i(\theta) = F(\theta) \cup \left(\bigcup_{j=1}^{i-1} \{x | f_j(x,\theta) \le f_j^*(\theta)\}\right) \end{array}$$

Theoretical foundation for Deep Financial Planning

- **We show the final feasible region $\boldsymbol{F_n(\theta)}$ is continuous correspondence**
  - Then, $F_n^*(\theta)$ is upper hemi-continuous

LEMMA 4.2 *Let $\mathcal{C}_1(\theta), \mathcal{C}_2(\theta), \ldots, \mathcal{C}_n(\theta)$ be continuous correspondence, then $\mathcal{C}(\theta) = \bigcup_{i=1}^n \mathcal{C}_i(\theta)$ is also continuous correspondence*

Proof. We first show that $\mathcal{C}$ is upper hemicontinuous. Let $\varepsilon > 0$ and $\theta_0 \in \Theta$ Since $\mathcal{C}_1(\theta), \mathcal{C}_2(\theta), \ldots, \mathcal{C}_n(\theta)$ are continuous, for each $\mathcal{C}_i(\theta), \exists \delta_i > 0$ such that

$$\theta \in \mathcal{B}_{\delta_i}(\theta_0) \Rightarrow \mathcal{C}_i(\theta) \subseteq \mathcal{B}_\varepsilon(\mathcal{C}_i(\theta_0)) \subseteq \mathcal{B}_\varepsilon(\mathcal{C}(\theta_0))$$

If we choose $\delta = \min_i \delta_i$

$$\theta \in \mathcal{B}_\delta(\theta_0) \Rightarrow \mathcal{C}_i(\theta) \subseteq \mathcal{B}_\varepsilon(\mathcal{C}_i(\theta_0)) \subseteq \mathcal{B}_\varepsilon(\mathcal{C}(\theta_0)), \forall i$$

Since $\mathcal{C}(\theta) = \bigcup_{i=1}^n \mathcal{C}_i(\theta)$, it means

$$\theta \in \mathcal{B}_\delta(\theta_0) \Rightarrow \bigcup_{i=1}^{n} \mathcal{C}_i(\theta) = \mathcal{C}(\theta) \subseteq \mathcal{B}_\varepsilon(\mathcal{C}(\theta_0))$$

Thus, $\mathcal{C}(\theta)$ is upper hemicontinuous. One can prove lower hemicontinuous with similar argument.

Theoretical foundation for Deep Financial Planning

THEOREM 4.3 *Supposed that $f_i(x,\theta)$ is continuous $\forall i = 1,2,\ldots,n$ and $\mathcal{F}(\theta)$ is continuous correspondence. Then, $\mathcal{F}_i(\theta)$ is continuous correspondence, $\forall i = 1,2,\ldots,n$*

Proof. We prove this theorem by mathematical induction. Since $\mathcal{F}_1(\theta) = \mathcal{F}(\theta)$, holds when $i = 1$. Suppose that theorem holds for $i = k$, and let $i = k+1$. Since $f_k(x,\theta)$ and $\mathcal{F}_k(\theta)$ are continuous, $f_k^*(\theta)$ is continuous from The Maximum Theorem. We now show that correspondence for goal constraint $\mathcal{G}_{k+1}(\theta) = \{x | f_k(x,\theta) \le f_k^*(\theta)\}$ is continuous. Let $g_k(x,\theta) = f_k(x,\theta) - f_k^*(\theta)$. Then, $\mathcal{G}_{k+1}(\theta) = \{x | g_k(x,\theta) \le 0\}$ and $g_k(x,\theta)$ is continuous. Let $\varepsilon > 0$ and $\theta_0 \in \Theta$ and let $\varepsilon' = \sup_{x \in \mathcal{B}_\varepsilon(\mathcal{G}_{k+1}(\theta_0))} g_k(x,\theta_0)$, then $\mathcal{B}_\varepsilon(\mathcal{G}_{k+1}(\theta_0)) = \{x | g_k(x,\theta_0) < \varepsilon'\}$. Since $g_k(x,\theta)$ is continuous, $\exists\delta$ such that

$$\theta \in \mathcal{B}_\delta(\theta_0) \Rightarrow g_k(x,\theta_0) - \varepsilon' < g_k(x,\theta) < g_k(x,\theta_0) + \varepsilon'$$

The left side indicates

$$g_k(x,\theta) \le 0 \Rightarrow g_k(x,\theta_0) < \varepsilon'$$

Thus, $\mathcal{G}_{k+1}(\theta) \subseteq \mathcal{B}_\varepsilon(\mathcal{G}_{k+1}(\theta_0))$ and $\mathcal{G}_{k+1}(\theta)$ is upper hemicontinuous by definition. Also, the right side indicates

$$g_k(x,\theta_0) \le 0 \Rightarrow g_k(x,\theta) < \varepsilon'$$

Thus, $\mathcal{G}_{k+1}(\theta_0) \subseteq \mathcal{B}_\varepsilon(\mathcal{G}_{k+1}(\theta))$ and $\mathcal{G}_{k+1}(\theta)$ is lower hemicontinuous by definition. Therefore, $\mathcal{G}_{k+1}(\theta)$ is continuous correspondence and $\mathcal{F}_{k+1}(\theta) = \mathcal{F}_k(\theta) \cup \mathcal{G}_{k+1}(\theta)$ is continuous by previous lemma. By mathematical induction, $\mathcal{F}_i(\theta)$ is continuous correspondence, $\forall i = 1,2,\ldots,n$ □

Theoretical foundation for Deep Financial Planning

- **Deep Financial Planning aim to approximate scenario-wise expectation**
  - We further prove the linear mapping preserve upper hemi-continuous

THEOREM 4.4 *Let T be linear transformation. If $\mathcal{C}(\theta)$ is upper hemicontinuous, $T(\mathcal{C}(\theta))$ is also upper hemicontinuous*

Proof. Write $T(x) = Ax + b$ and let $\varepsilon > 0$ and $\varepsilon' = \varepsilon/\|A\|$ where $\|A\|$ is norm of A Since $\mathcal{C}(\theta)$ is upper hemicontinuous, $\exists \delta$ such that

$$\theta \in \mathcal{B}_\delta(\theta_0) \Rightarrow \mathcal{C}(\theta) \subseteq \mathcal{B}'_\varepsilon(\mathcal{C}(\theta_0))$$

If $x \in \mathcal{B}'_\varepsilon(\mathcal{C}(\theta_0))$, $\|x - y\| < \varepsilon', \forall y \in \mathcal{C}(\theta_,)$. Thus, $\|Ax - Ay\| \leq \|A\|\|x - y\| < \varepsilon'\|A\|$ This implies,

$$T(\mathcal{C}(\theta)) \subseteq T(\mathcal{B}_{\varepsilon'}(\mathcal{C}(\theta_0))) \subseteq \mathcal{B}_{\varepsilon'\|A\|}(T(\mathcal{C}(\theta_0))) = \mathcal{B}_\varepsilon(T(\mathcal{C}(\theta_0)))$$

Thus, $T(\mathcal{C}(\theta))$ is upper hemicontinuous □

Empirical Results: Simple ALM

- **Train data: 100,000; Test data: 10,000**

Target	$\boldsymbol{R^2}$
$E_s[a_{1,1,s}]$	0.9988
$E_s[a_{2,1,s}]$	0.9988
$E_s[a_{1,2,s}]$	0.9995
$E_s[a_{2,2,s}]$	0.9995
$E_s[a_{1,3,s}]$	0.9998
$E_s[a_{2,3,s}]$	0.9998
$E_s[c_s]$	0.9998

Empirical Results: Simple GBI

- **Train data: 1,152,000; Test data: 144,000**

Target	R^2
$E_s[a_{1,1,s}]$	0.9989
$E_s[a_{2,1,s}]$	0.9989
$E_s[a_{1,2,s}]$	0.9996
$E_s[a_{2,2,s}]$	0.9993
$E_s[a_{1,3,s}]$	1.0000
$E_s[a_{2,3,s}]$	1.0000

Target	R^2
$E_s[g^1_{\sigma_1,s}]$	0.9938
$E_s[g^2_{\sigma_2,s}]$	1.0000
$E_s[g^3_{\sigma_3,s}]$	1.0000
$E_s[c^1_{\sigma_1,s}]$	0.9992
$E_s[c^2_{\sigma_2,s}]$	0.9996
$E_s[c^3_{\sigma_3,s}]$	0.9997

Empirical Results: Advanced GBI

- **Train data: 4,608,000; Test data: 576,000**

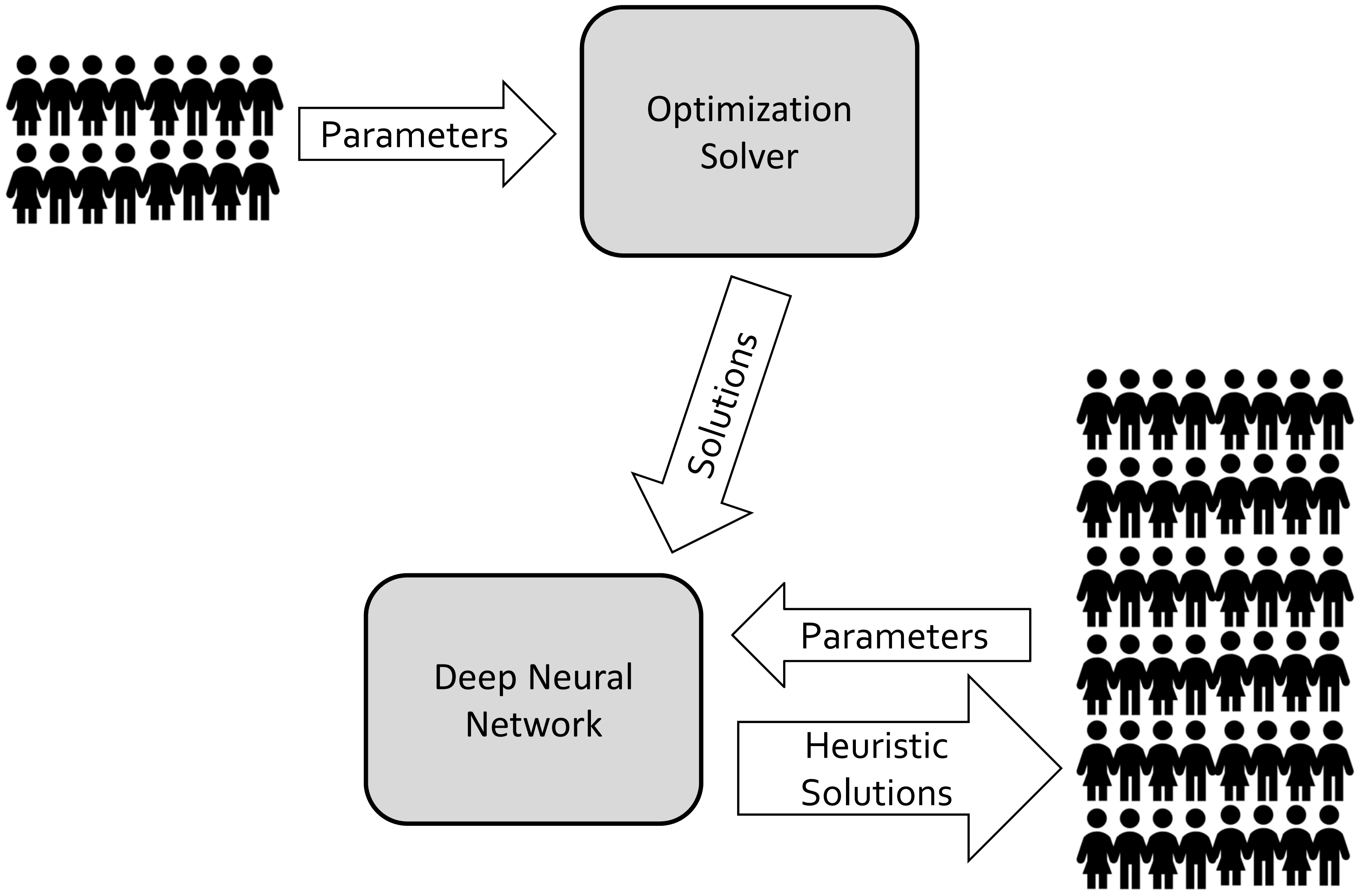
Target	R^2
$E_s[a_{1,1,s}]$	0.9950
$E_s[a_{2,2,s}]$	0.9954
$E_s[a_{3,1,s}]$	0.9992
$E_s[a_{1,2,s}]$	0.9933
$E_s[a_{2,2,s}]$	0.9954
$E_s[a_{3,2,s}]$	0.9990

Target	R^2
$E_s[a_{2,3,s}]$	0.9995
$E_s[a_{3,3,s}]$	0.9994
$E_s[a_{2,4,s}]$	0.9995
$E_s[a_{3,4,s}]$	0.9997
$E_s[g^1_{\sigma_1,s}]$	0.9978
$E_s[g^2_{\sigma_2,s}]$	0.9999

Target	R^2
$E_s[g^3_{\sigma_3,s}]$	0.9999
$E_s[c^1_{\sigma_1,s}]$	0.9988
$E_s[c^2_{\sigma_2,s}]$	0.9993
$E_s[c^3_{\sigma_3,s}]$	0.9986

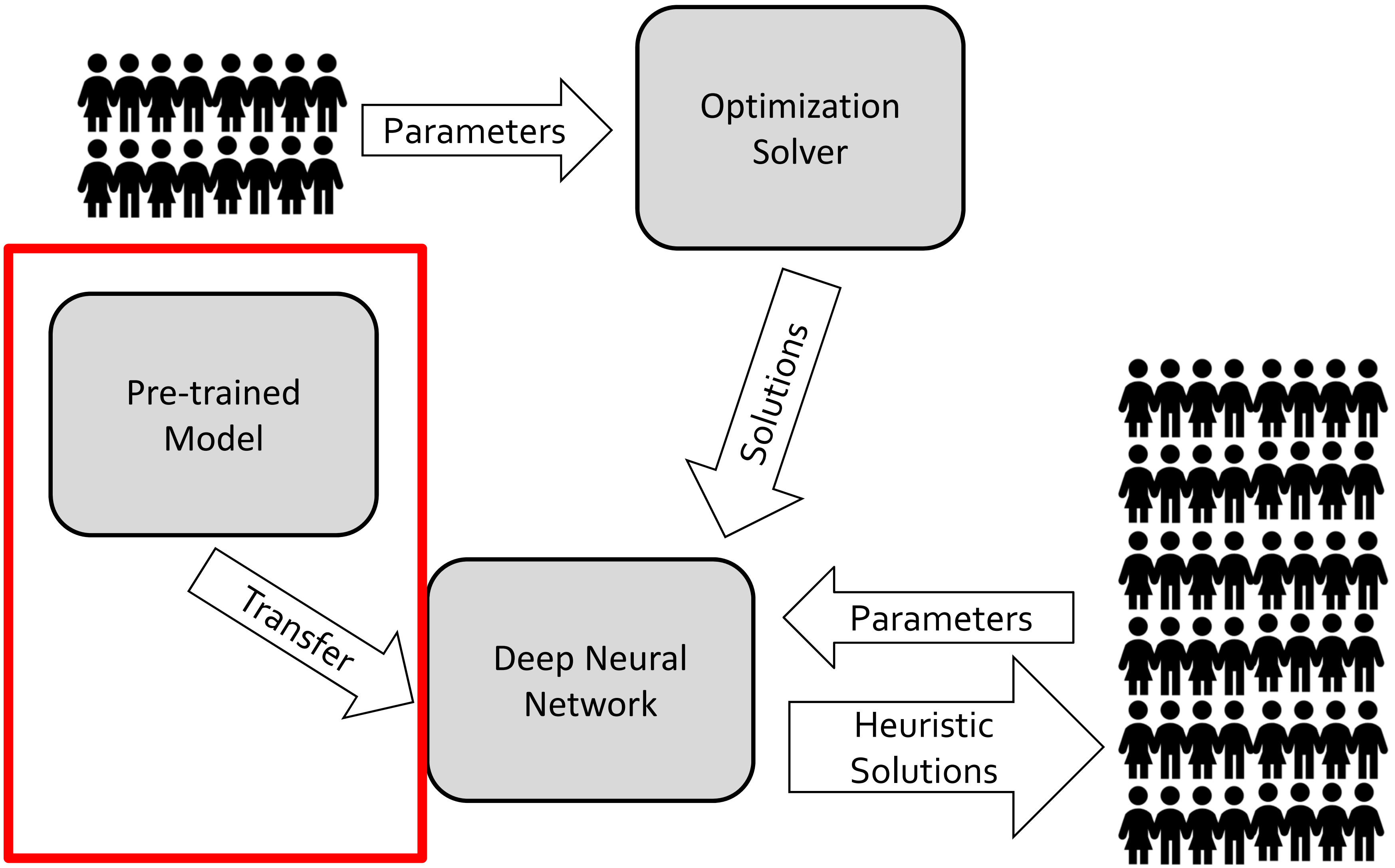
Parametric Optimization: NN-Based Approach

- **Neural Network (NN) can provide heuristics for decision-making**



Parametric Optimization: NN-Based Approach

- **Neural Network (NN) can provide heuristics for decision-making**



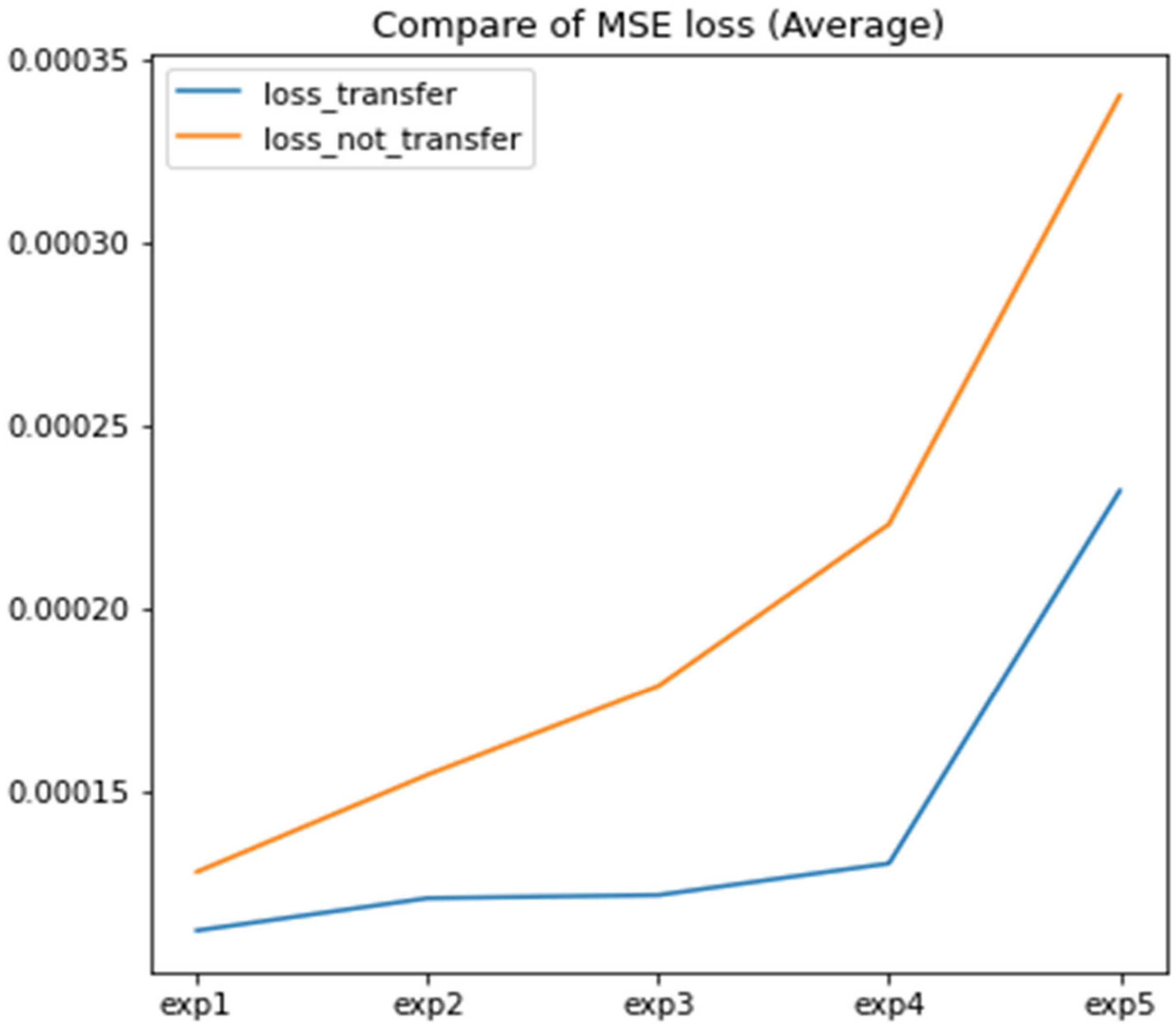
Empirical Results: Transfer learning

- **It is possible to transfer information from small-sized financial planning problems to large ones?**

- **We considered two identical Advanced GBI problems except for the number of scenarios**
  - We repeat training and testing 100 times independently and compared the average MSE loss with and without transfer learning for datasets of five different sizes

	With Transfer learning		**Without transfer learning**
	Big-size	**Small-size**	**Big-size**
# of scenarios	3360	200	3360
# of Training data (Exp 1)	173,250,	9,000,000	192,000
# of Training data (Exp 2)	138,600,	7,200,000	153,600
# of Training data (Exp 3)	103,950,	5,400,000	115,200
# of Training data (Exp 4)	69,300,	3,600,000	76,800
# of Training data (Exp 5)	34,650,	1,800,000	38,400
# of Test data	24,000	-	24,000

Empirical Results: Transfer learning



Remaining Questions

- What's the proper architecture of NN for DFP?
- Any better way for transfer learning?
- Does transfer learning work for general parametric optimization?
- If so, does it make sense to construct a pretrained model (like VGG-16, ResNet50, DenseNet12, InceptionV3, Xception, etc.) for PO?
- If not, can we build a pretrained model for specific tasks such as financial planning, production management, energy planning, etc.?

  => Given what we have observed, this might be a good bet.

Thank you for listening and enjoy the lunch!