

---

# Machine Learning Driven Heuristics for Combinatorial Optimization Problems

Core Algorithms

---



2024.05.17

Data Mining & Quality Analytics Lab. Open Seminar

차민성

# 발표자 소개



## ❖ 차민성 (Minsung Cha)

- 고려대학교 일반대학원 산업경영공학과 재학
- Data Mining & Quality Analytics Lab.
- 지도교수 : 김성범 교수님

## ❖ 관심 연구 분야

- Optimization
- Deep Learning Algorithms
- Reinforcement Learning Algorithms

## ❖ 연락처

- [djpanda1217@korea.ac.kr](mailto:djpanda1217@korea.ac.kr)

# Table of Contents

- 1 What is Optimization?
- 2 Combinatorial Optimization Problems
- 3 Sequence Modeling for TSP
- 4 Pointer Networks
- 5 Neural Combinatorial Optimization with Reinforcement Learning
- 6 Attention, Learn to Solve Routing Problems!
- 7 Conclusion
- 8 References

ML Driven Heuristics for Combinatorial Optimization Problems - Core Algorithms

# What Is Optimization?

- Analogy with Quadratic Function

# Optimization Problems

최적화 문제란?



## ChatGPT

세미나를 준비하시는 것을 축하드립니다! '최적화 문제'를 가볍게 설명해 드릴게요.

최적화 문제란 주어진 조건 속에서 가장 좋은 해결책을 찾는 수학적 문제입니다. 이 문제는 보통 어떤 목적 함수(목표를 나타내는 함수)를 최대 또는 최소화하는 변수들의 조합을 찾는 것을 목표로 합니다.

간단한 예시로 설명해 보면, 예를 들어 어떤 제품을 만들 때 가격과 품질 사이의 균형점을 찾는 것이 최적화 문제의 일종일 수 있습니다. 이때 가격과 품질이라는 변수를 조절하여 손익이 최대화되거나 품질이 최대화되는 상황을 고려할 수 있습니다.

이런 최적화 문제는 다양한 분야에서 사용됩니다. 예를 들어 공학에서는 자동차나 비행기의 구조 최적화, 경제학에서는 이익을 극대화하는 소비량의 결정, 물리학에서는 우주 정거장의 궤도 최적화 등 다양한 분야에서 활용됩니다.

이런 식으로 최적화 문제는 주어진 제약 조건 하에서 최대 또는 최소 값을 찾는 문제로 정의됩니다. 세미나에서 이 개념을 소개하실 때, 좀 더 구체적인 예시나 응용 분야를 들어 설명하시면 이해도가 높아질 것입니다. 최적화 문제는 수학적으로 해결 방법이 다양하게 연구되고 있어서, 세미나에서는 이런 다양한 접근 방법에 대해 간략히 언급하는 것도 도움이 될 것입니다.



Thank you, ChatGPT!

# Optimization Problems

최적화 문제란?

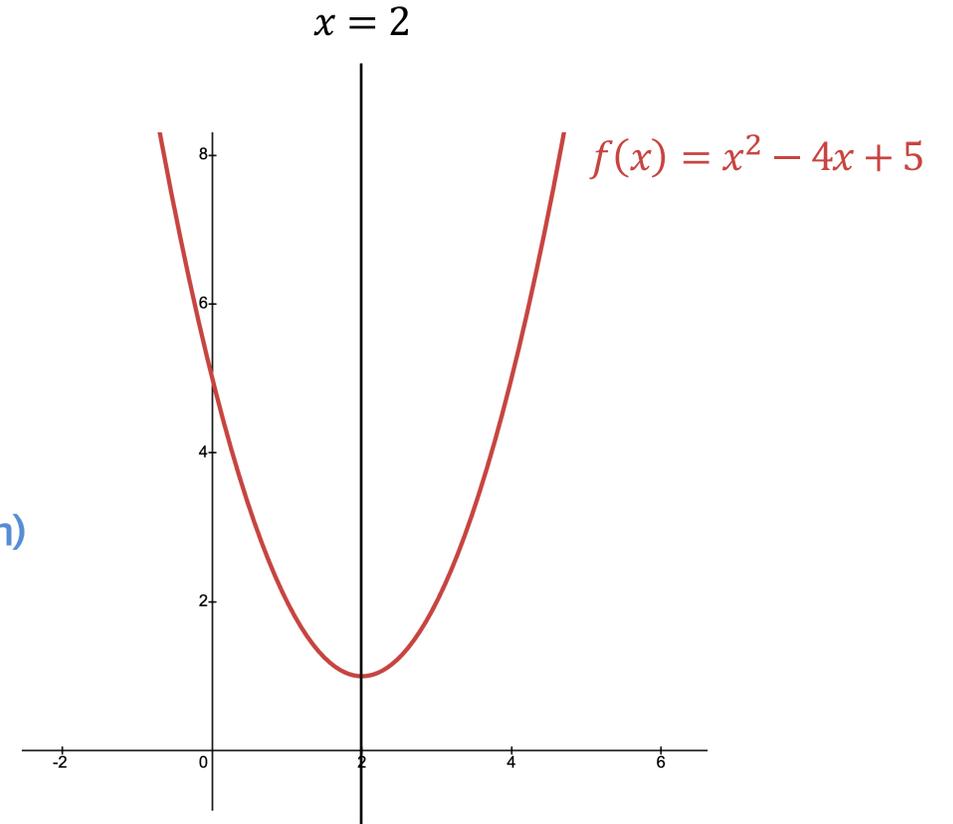
❖ 최적화 문제는 다음과 같이 구성됩니다!

- 가장 좋은 해결책(최적해, Optimal Solution)을 찾자! - **목적함수(Objective Function)**를 최대화/최소화하는
- 주어진 조건 하에서 - **제약조건(Constraint)**

Ex) 이차함수의 값이 최소가 되게 하는  $x$ 의 값을 찾는 문제 (정의역 실수)

**Q.** 실수 전체 범위에서,  
함수  $f(x) = x^2 - 4x + 5$ 의 함수값이  
최소가 되게 하는  $x$ 의 값을 찾으시오.  
목적함수(Objective Function)

**A.**  $x^* = 2$   
최적해(Optimal Solution)



# Optimization Problems

최적화 문제란?

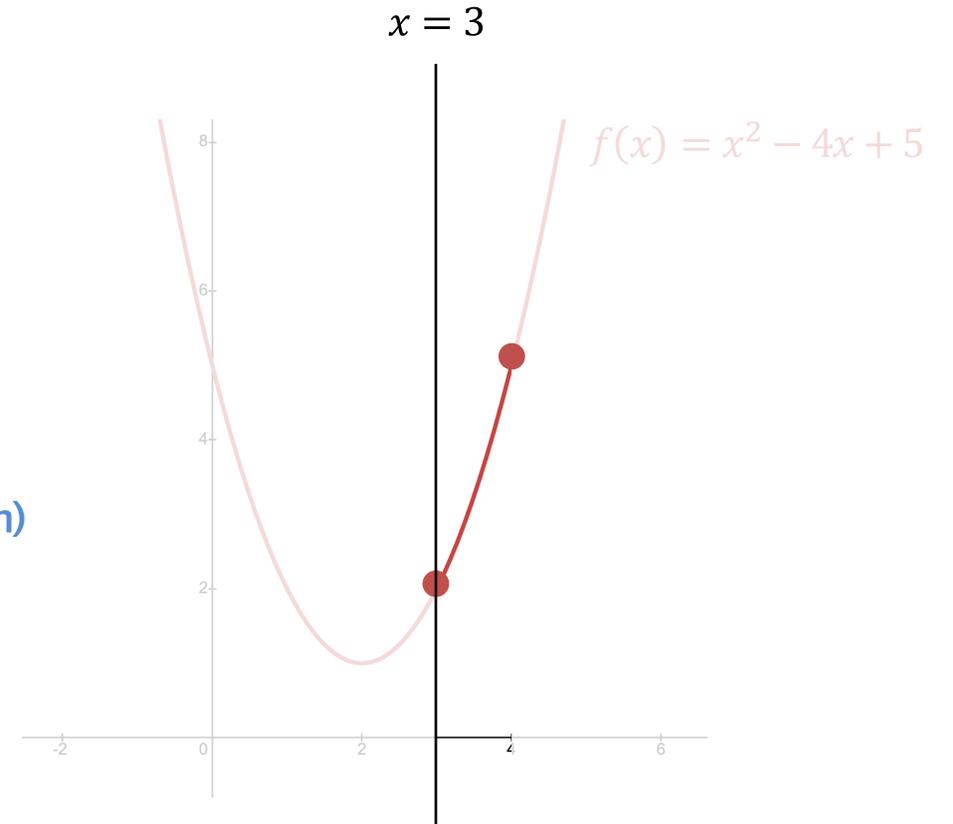
❖ 최적화 문제는 다음과 같이 구성됩니다!

- 가장 좋은 해결책(최적해, Optimal Solution)을 찾자! - **목적함수(Objective Function)**를 최대화/최소화하는
- 주어진 조건 하에서 - **제약조건(Constraint)**

Ex) 이차함수의 값이 최소가 되게 하는  $x$ 의 값을 찾는 문제 (정의역 실수, 범위 제한)

**Q.**  $x \in [3, 4]$ 일 때, **제약조건(Constraint)**  
함수  $f(x) = x^2 - 4x + 5$ 의 함수값이 **목적함수(Objective Function)**  
최소가 되게 하는  $x$ 의 값을 찾으시오.

**A.**  $x^* = 3$  **최적해(Optimal Solution)**



# Optimization Problems

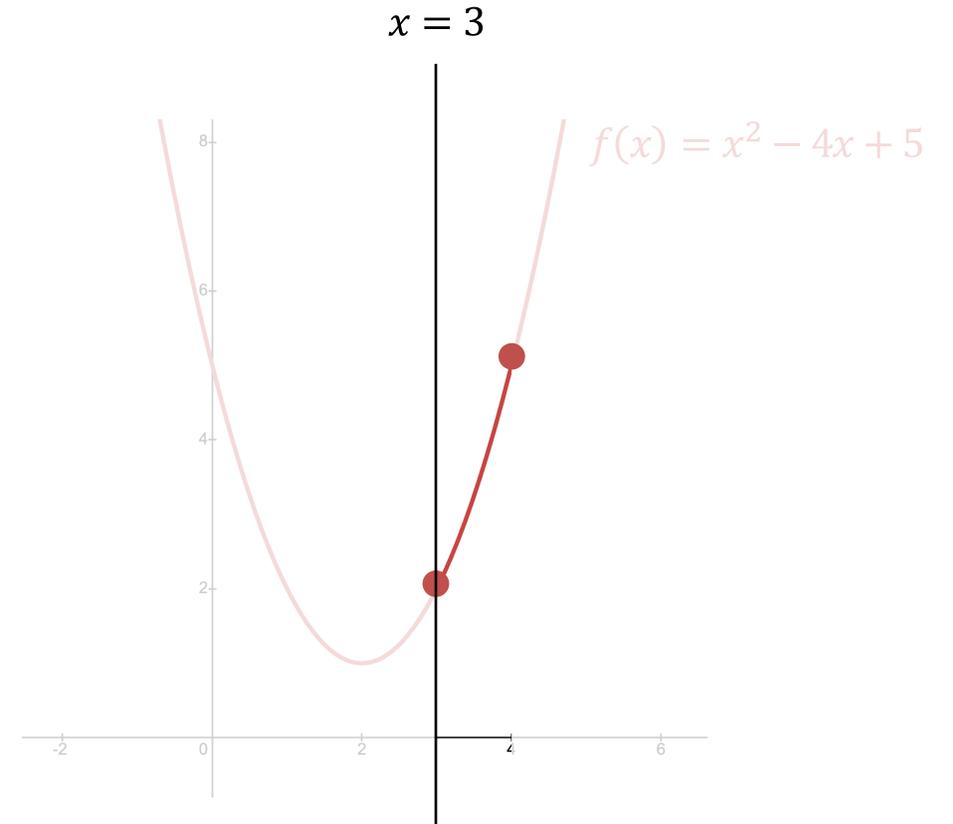
최적화 문제란?

❖ 최적화 문제는 다음과 같이 구성됩니다!

- 가장 좋은 해결책(최적해, Optimal Solution)을 찾자! - **목적함수(Objective Function)**를 최대화/최소화하는
- 주어진 조건 하에서 - **제약조건(Constraint)**

Ex) 이차함수의 값이 최소가 되게 하는  $x$ 의 값을 찾는 문제 (정의역 실수, 범위 제한)

$$\operatorname{argmin}_{3 \leq x \leq 4} x^2 - 4x + 5 = 3$$



# Optimization Problems

최적화 문제란?

❖ 최적화 문제는 다음과 같이 구성됩니다!

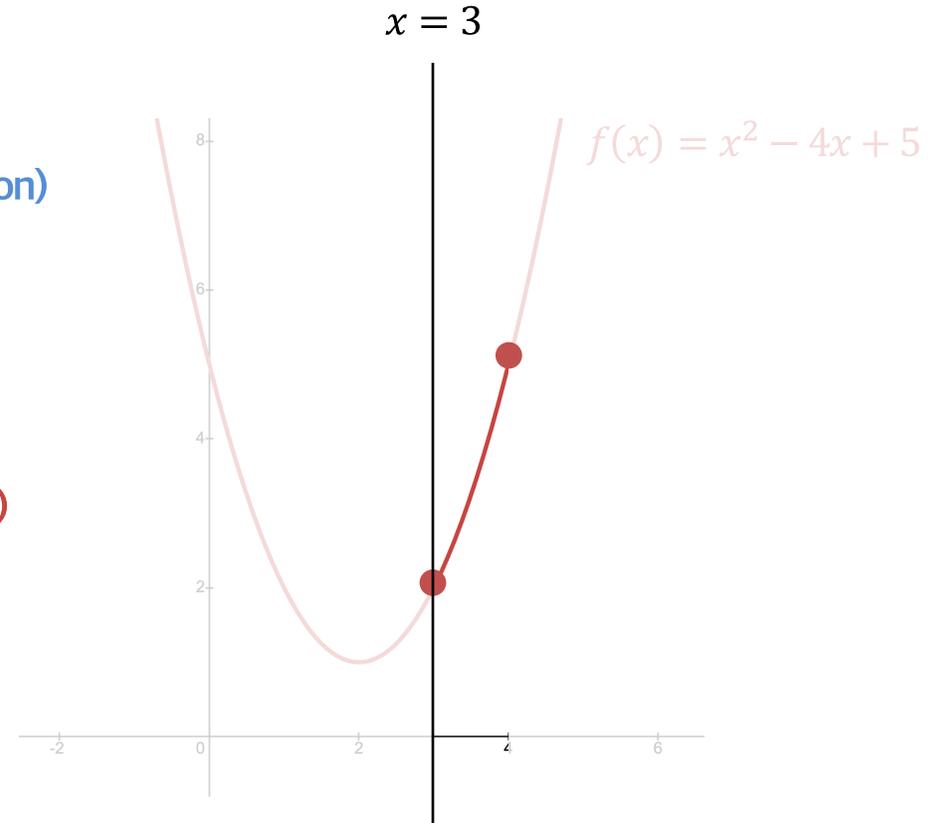
- 가장 좋은 해결책(최적해, **Optimal Solution**)을 찾자! - **목적함수(Objective Function)**를 최대화/최소화하는
- 주어진 조건 하에서 - **제약조건(Constraint)**

Ex) 이차함수의 값이 최소가 되게 하는  $x$ 의 값을 찾는 문제 (정의역 실수, 범위 제한)

**Q.** minimize  $f(x) = x^2 - 4x + 5$   
s. t.  $3 \leq x \leq 4$

목적함수(Objective Function)  
제약조건(Constraint)

**A.**  $x^* = 3$   
최적해(Optimal Solution)



완전제곱식 활용  
미분  
구간 양끝 값 비교  
...

# Optimization Problems

정의역이 실수가 아니라면?

❖ 최적화 문제는 다음과 같이 구성됩니다!

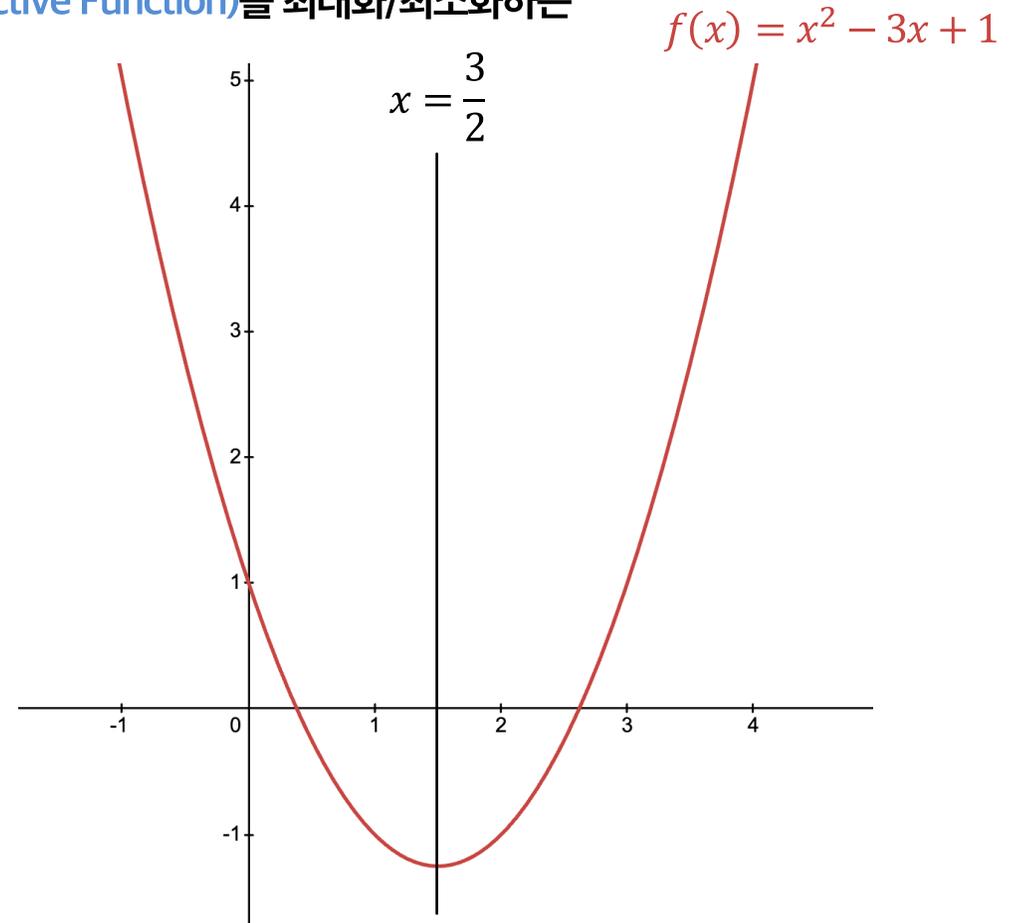
- 가장 좋은 해결책(최적해, Optimal Solution)을 찾자! - **목적함수(Objective Function)**를 최대화/최소화하는
- 주어진 조건 하에서 - **제약조건(Constraint)**

Ex) 이차함수의 값이 최소가 되게 하는  $x$ 의 값을 찾는 문제 (정의역 실수)

**Q.** 
$$\begin{aligned} & \text{minimize} && f(x) = x^2 - 3x + 1 \\ & \text{s. t.} && x \in \mathbb{R} \end{aligned}$$

**A.** 
$$x^* = \frac{3}{2}$$

완전제곱식 활용  
미분  
구간 양끝 값 비교  
...



Thank you, Desmos!

# Optimization Problems

정의역이 실수가 아니라면?

❖ 최적화 문제는 다음과 같이 구성됩니다!

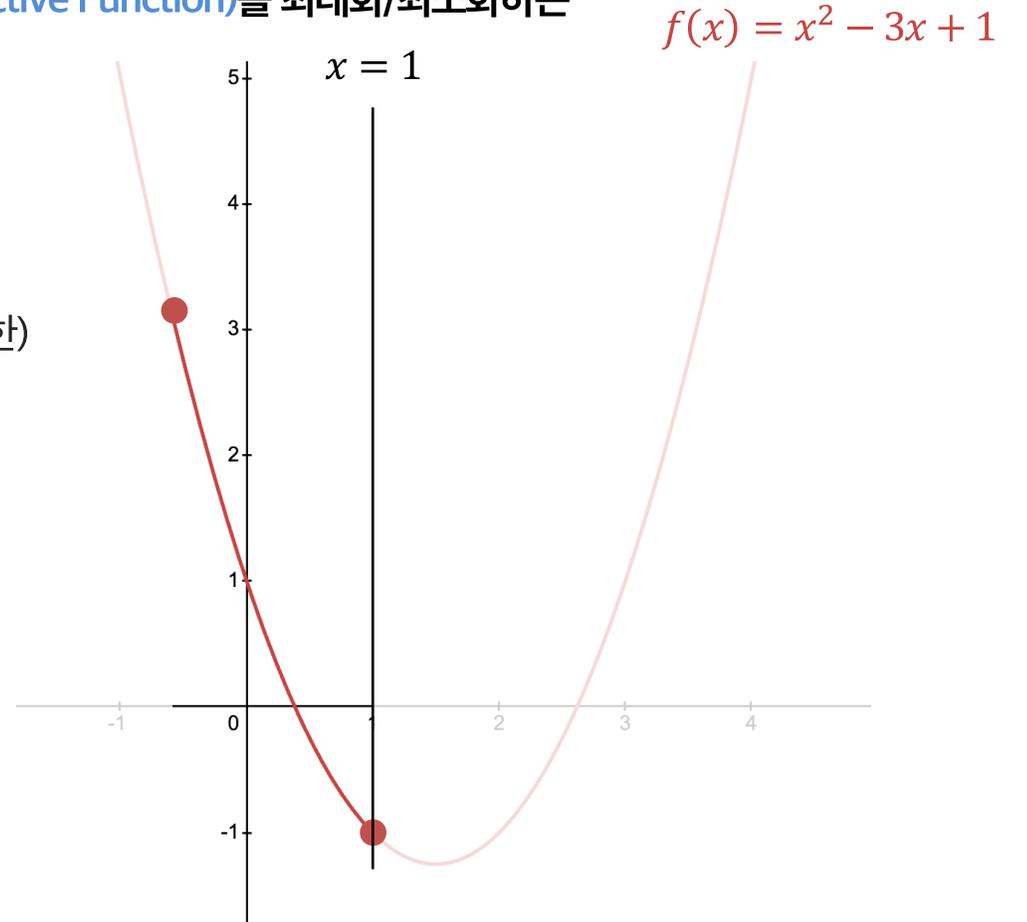
- 가장 좋은 해결책(최적해, Optimal Solution)을 찾자! - **목적함수(Objective Function)**를 최대화/최소화하는
- 주어진 조건 하에서 - **제약조건(Constraint)**

Ex) 이차함수의 값이 최소가 되게 하는  $x$ 의 값을 찾는 문제 (정의역 실수, 범위 제한)

**Q.** 
$$\begin{aligned} & \text{minimize } f(x) = x^2 - 3x + 1 \\ & \text{s.t. } x \in \left[-\frac{1}{2}, 1\right] \end{aligned}$$

**A.**  $x^* = 1$

완전제곱식 활용  
미분  
구간 양끝 값 비교  
...



Thank you, Desmos!

# Optimization Problems

정의역이 실수가 아니라면?

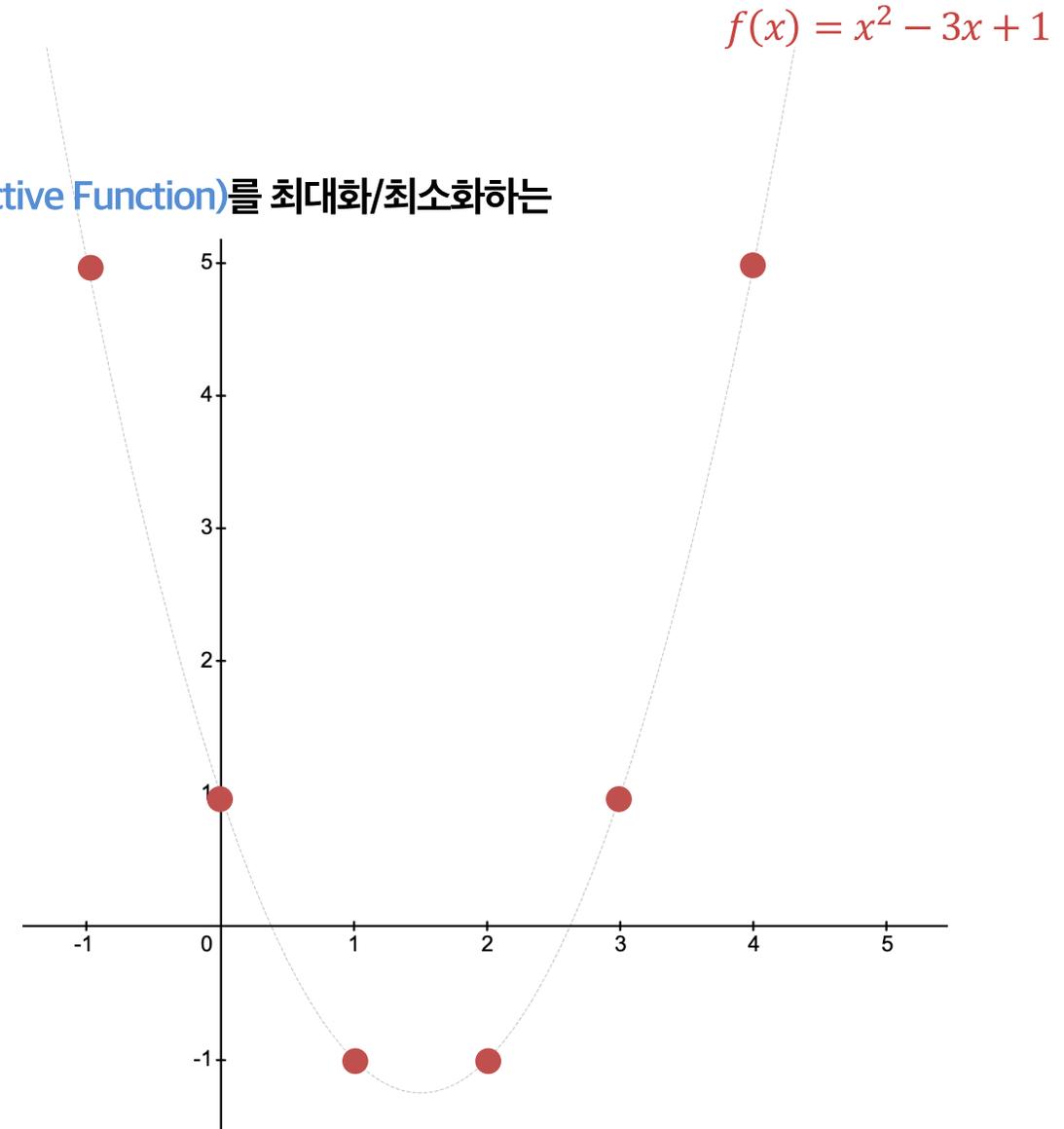
❖ 최적화 문제는 다음과 같이 구성됩니다!

- 가장 좋은 해결책(최적해, Optimal Solution)을 찾자! - 목적함수(Objective Function)를 최대화/최소화하는
- 주어진 조건 하에서 - 제약조건(Constraint)

Ex) 이차함수의 값이 최소가 되게 하는  $x$ 의 값을 찾는 문제

$$\begin{array}{ll} \text{Q.} & \text{minimize } f(x) = x^2 - 3x + 1 \\ & \text{s.t. } x \in \mathbb{Z} \end{array}$$

~~완전체공식 활용  
미분  
구간 양끝 값 비교  
...~~



Thank you, Desmos!

ML Driven Heuristics for Combinatorial Optimization Problems - Core Algorithms

# Combinatorial Optimization Problems

# Optimization Problems

최적화 문제란?

❖ 최적화 문제는 다음과 같이 구성됩니다!

- 가장 좋은 해결책(최적해, Optimal Solution)을 찾아! - **목적함수(Objective Function)**를 최대화/최소화하는
- 주어진 조건 하에서 - **제약조건(Constraint)**



알겠습니다. 문장을 조금 수정하여 제품 생산과 관련된 내용을 다시 표현해 보겠습니다.

예시 LP 문제:

한 공장에서 세 가지 원료를 사용하여 제품을 생산하려고 합니다. 이 공장은 원료 A, 원료 B, 원료 C를 사용하여 제품을 생산합니다. 각 제품의 생산에는 다음과 같은 원료 소모가 필요합니다.

- 제품 X를 1리터 생산하기 위해서는 2리터의 원료 A와 1리터의 원료 B가 필요합니다.
- 제품 Y를 1리터 생산하기 위해서는 1리터의 원료 A와 3리터의 원료 C가 필요합니다.

제품 X와 제품 Y의 이익은 각각 10달러, 15달러입니다. 공장은 각 원료의 재고량에 따라 제품을 생산할 수 있습니다. 원료 A, B, C의 최대 재고량은 각각 100리터, 80리터, 120리터입니다.

대표적인 최적화 문제인 LP의 예시

# Optimization Problems

최적화 문제란?

❖ 최적화 문제는 다음과 같이 구성됩니다!

- **가장 좋은 해결책(최적해, Optimal Solution)**을 찾자! - **목적함수(Objective Function)**를 최대화/최소화하는
- **주어진 조건 하에서 - 제약조건(Constraint)**

알겠습니다. 문장을 조금 수정하여 제품 생산과 관련된 내용을 다시 표현해 보겠습니다.

예시 LP 문제:

한 공장에서 세 가지 원료를 사용하여 제품을 생산하려고 합니다. 이 공장은 원료 A, 원료 B, 원료 C를 사용하여 제품을 생산합니다. 각 제품의 생산에는 다음과 같은 원료 소모가 필요합니다.

- 제품 X를 1리터 생산하기 위해서는 2리터의 원료 A와 1리터의 원료 B가 필요합니다.
- 제품 Y를 1리터 생산하기 위해서는 1리터의 원료 A와 3리터의 원료 C가 필요합니다.

제품 X와 제품 Y의 이익은 각각 10달러, 15달러입니다. 공장은 각 원료의 재고량에 따라 제품을 생산할 수 있습니다. 원료 A, B, C의 최대 재고량은 각각 100리터, 80리터, 120리터입니다.

## 문제 상황:

어떤 회사가 두 종류의 제품(제품 1, 제품 2)을 생산하려고 한다. 제품 생산에 사용되는 원료는 세 종류(원료 A, 원료 B, 원료 C)이다. 각 제품 1리터를 생산하기 위해 필요한 원료 소모는 다음과 같다.

**제품 1을 생산하기 위해서는 1리터 당 원료 A 2리터, 원료 B 1리터, 원료 C 1리터가 필요하다.**

**제품 2를 생산하기 위해서는 1리터 당 원료 A 1리터, 원료 B 3리터가 필요하다.**

제품 1 생산 시의 이익은 1리터 당 10달러, 제품 2 생산 시의 이익은 1리터 당 15달러이다.

**원료 A, B, C의 재고량이 각각 100리터, 80리터, 120리터로 제한되었다고 할 때,**

**제품 1, 2를 각각 몇 리터씩 생산해야 전체 이익 Z를 최대화할 수 있겠는가?**

\*언급된 사항 외의 비용 및 제한사항은 존재하지 않는 것으로 간주한다.

# Optimization Problems

최적화 문제란?

❖ 최적화 문제는 다음과 같이 구성됩니다!

- 가장 좋은 해결책(최적해, **Optimal Solution**)을 찾자! - **목적함수(Objective Function)**를 최대화/최소화하는
- 주어진 조건 하에서** - **제약조건(Constraint)**

문제 상황:

어떤 회사가 두 종류의 제품(제품 1, 제품 2)을 생산하려고 한다. 제품 생산에 사용되는 원료는 세 종류(원료 A, 원료 B, 원료 C)이다. 각 제품 1리터를 생산하기 위해 필요한 원료 소모는 다음과 같다.

제품 1을 생산하기 위해서는 1리터 당 원료 A 2리터, 원료 B 1리터, 원료 C 1리터가 필요하다.

제품 2를 생산하기 위해서는 1리터 당 원료 A 1리터, 원료 B 3리터가 필요하다.

제품 1 생산 시의 이익은 1리터 당 10달러, 제품 2 생산 시의 이익은 1리터 당 15달러이다.

원료 A, B, C의 재고량이 각각 100리터, 80리터, 120리터로 제한되었다고 할 때,

제품 1, 2를 각각 몇 리터씩 생산해야 전체 이익 Z를 최대화할 수 있겠는가?

\*언급된 사항 외의 비용 및 제한사항은 존재하지 않는 것으로 간주한다.

$$\text{maximize } Z = 10x_1 + 15x_2$$

s. t.

$$2x_1 + x_2 \leq 100$$

$$x_1 + 3x_2 \leq 80$$

$$x_1 \leq 120$$

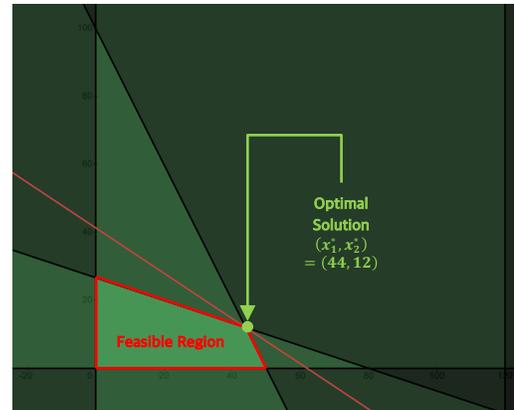
$$x_1, x_2 \geq 0$$

$$x_1, x_2 \in \mathbb{R}$$

$$x_1^* = 44, x_2^* = 12$$

$x_1$ : 제품 1의 생산량 (리터)

$x_2$ : 제품 2의 생산량 (리터)



Simplex Method  
Interior Point Algorithm

...

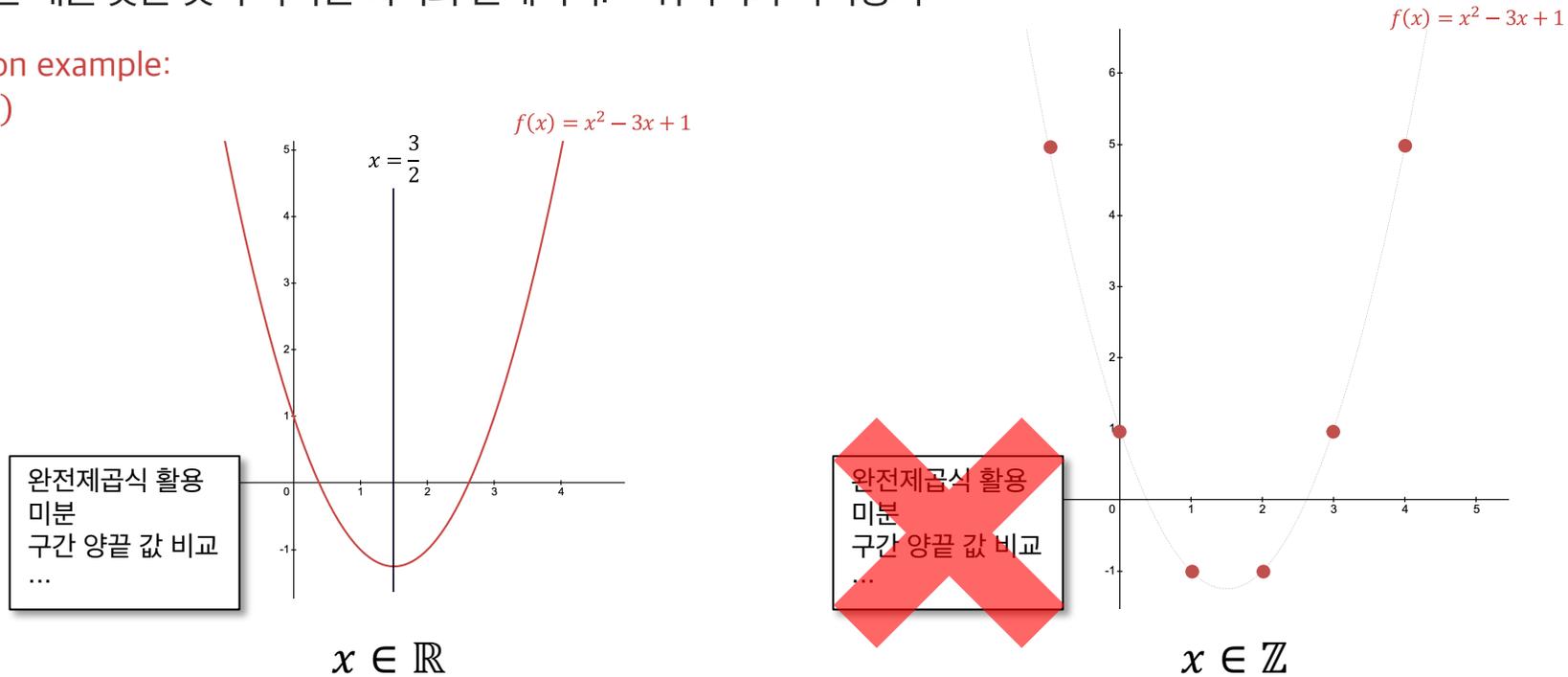
Thank you, Desmos!

# Combinatorial Optimization Problems

조합최적화 문제란?

- ❖ 기본적으로는 최적화 문제의 한 범주이지만,
  - 어떤 ‘조합’이 문제의 목적함수를 최대화/최소화하는 데에 가장 좋은가?
  - “조합 최적화의 영역은 가능해가 이산 집합에 속하거나 이산적인 것으로 변환될 수 있고, 가장 좋은 해를 찾는 것이 목적인 최적화 문제이다.” - 위키백과 약식정의

Revisit - quadratic function example:  
*minimize  $f(x)$*



# Combinatorial Optimization Problems

조합최적화 문제란?

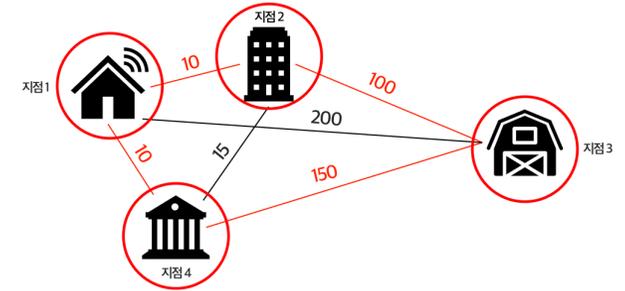
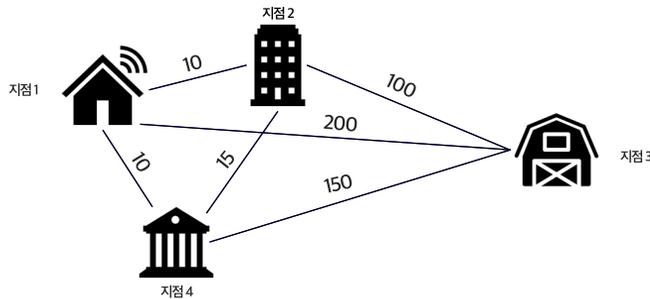
❖ 기본적으로는 최적화 문제의 한 범주이지만,

- 어떤 ‘조합’이 문제의 목적함수를 최대화/최소화하는 데에 가장 좋은가?
- “조합 최적화의 영역은 가능해가 이산 집합에 속하거나 이산적인 것으로 변환될 수 있고, 가장 좋은 해를 찾는 것이 목적인 최적화 문제이다.” - 위키백과 약식정의

조합최적화의 대표적 예시: TSP 문제

문제 상황:

$k$ 개의 지점을 각각 한 번씩만 방문하고 처음 시작한 지점으로 돌아오는 cycle을 찾고자 한다. 각 지점 사이의 거리를  $d_{ij}$ 라고 할 때, cycle의 전체 길이  $Z$ 가 최소가 되도록 하려면 방문 순서는 어떻게 되어야 하는가?



minimize

$$Z = \sum_{i=1}^k \sum_{j=1}^k d_{ij} x_{ij}$$

s. t.

$$\sum_{i=1}^k x_{ij} = 1$$

$$\sum_{j=1}^k x_{ij} = 1$$

$x_{ij}$  :  
 $i$ 번 지점과  $j$ 번 지점을 연결하는 간선을 사용하는지 여부

$$u_i - u_j + kx_{ij} \leq k - 1 \quad \forall i, j \in V, i \neq j$$

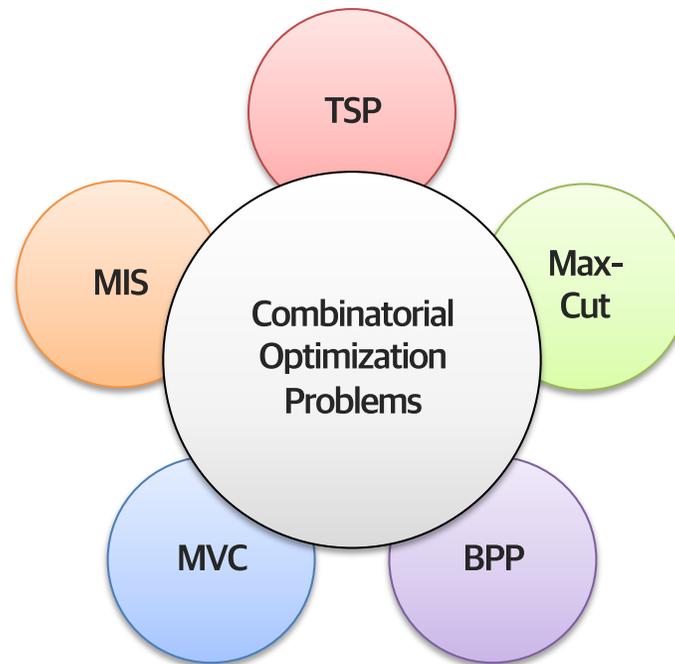
$$x_{ij} \in \{0,1\} \quad \forall i, j \in V$$

$$u_i \in \{0,1\} \quad \forall i \in V$$

# Combinatorial Optimization Problems

## 조합최적화 문제의 특징

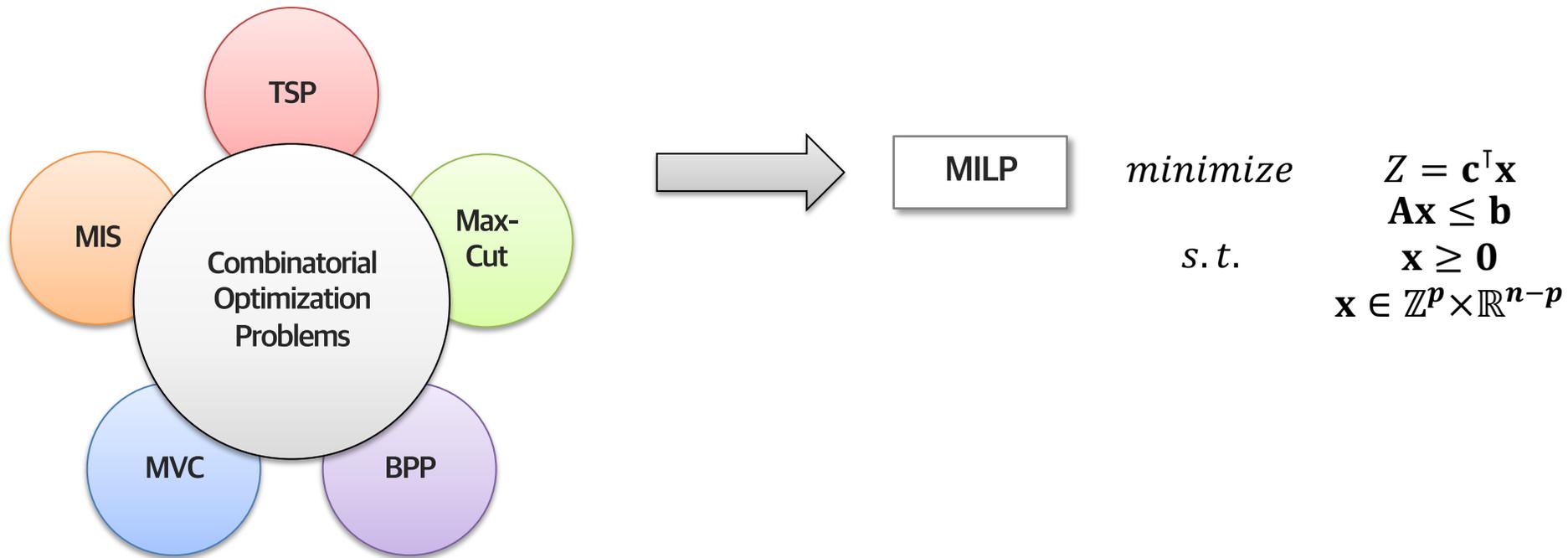
- **조합최적화(Combinatorial Optimization)** 문제들은 TSP, MIS, Max-Cut, BPP, MVC, MIS 등의 세부 클래스로 나뉨
- 현실문제를 그대로 모델링 가능하다는 측면에서 중요한 문제
- 대부분 MILP(Mixed Integer Linear Programming)로 모델링 가능



# Combinatorial Optimization Problems

조합최적화 문제의 특징

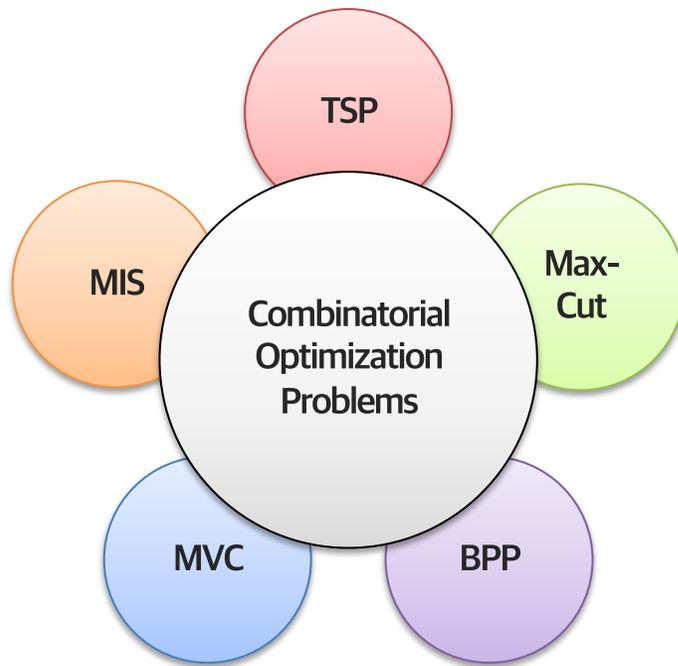
- 조합최적화(Combinatorial Optimization) 문제들은 TSP, MIS, Max-Cut, BPP, MVC, MIS 등의 세부 클래스로 나뉨
- 현실문제를 그대로 모델링 가능하다는 측면에서 중요한 문제
- 대부분 MILP(Mixed Integer Linear Programming)로 모델링 가능



# Combinatorial Optimization Problems

조합최적화 문제의 특징

- 조합최적화(Combinatorial Optimization) 문제들은 TSP, MIS, Max-Cut, BPP, MVC, MIS 등의 세부 클래스로 나뉨
- 현실문제를 그대로 모델링 가능하다는 측면에서 중요한 문제
- 대부분 MILP(Mixed Integer Linear Programming)로 모델링 가능
- 대부분 NP-hard → 문제 크기가 커지면 현실적인 시간 내에 Optimal Solution 획득 불가 → Heuristic 필요



MILP

*minimize*

$$Z = \mathbf{c}^T \mathbf{x}$$

$$\mathbf{Ax} \leq \mathbf{b}$$

*s. t.*

$$\mathbf{x} \geq \mathbf{0}$$

$$\mathbf{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p}$$

- NP-hard
- 문제 크기가 커지면 현실적인 시간 내에 Optimal Solution 획득 불가 → Heuristic

# Machine Learning Driven Heuristics

머신러닝 기반 Heuristics의 등장 배경

## ❖ Heuristic

- 현실적인 시간 내에 optimal solution을 구할 수 없는 조합최적화 문제에 대하여, 빠른 시간 내에 (sub)optimal solution을 찾을 수 있도록 하는 방법론

## ❖ 사람이 직접 만드는 hand-crafted heuristics의 한계점

- 문제에 대한 깊은 도메인 지식이 필요하거나
- 문제 구조 등에 민감(문제의 세부 조건 변화에 따라 그 성능이 크게 변화)

→ 머신러닝 기반 Heuristics!

# Machine Learning Driven Heuristics

머신러닝 기반 Heuristics의 등장 배경

## ❖ Heuristic

- 현실적인 시간 내에 optimal solution을 구할 수 없는 최적화 문제에 대하여 빠른 시간 내에 (sub)optimal solution을 찾을 수 있도록 하는 방법론

## ❖ 사람이 직접 만드는 hand-crafted heuristics의 한계점

- 깊은 도메인 지식이 필요함
- 문제 구조 등에 민감(문제의 구조 변화에 따라 성능이 크게 변화)

→ 머신러닝 기반 Heuristics!

# Traveling Salesman Problem (외판원 문제)

ML Driven Heuristics for Combinatorial Optimization Problems - Core Algorithms

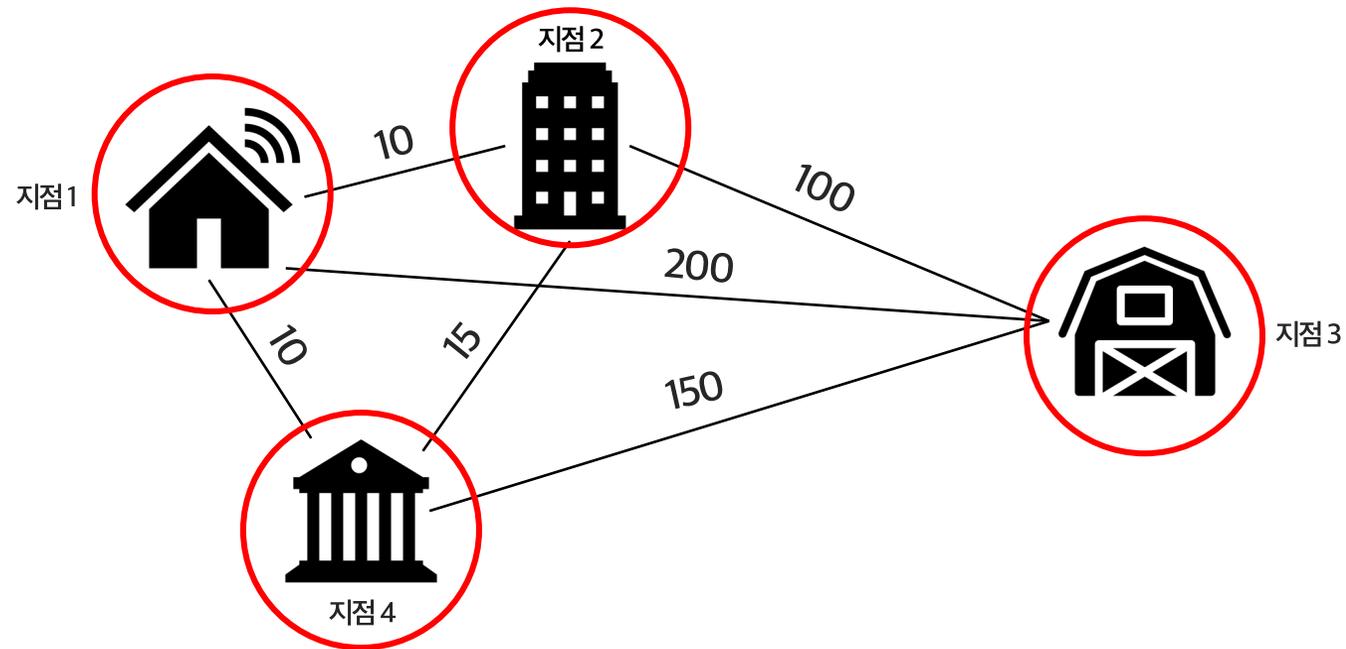
# Sequence Modeling for TSP

# Traveling Salesman Problem

조합최적화 문제의 대표적인 예시

## ❖ TSP (Traveling Salesman Problem)

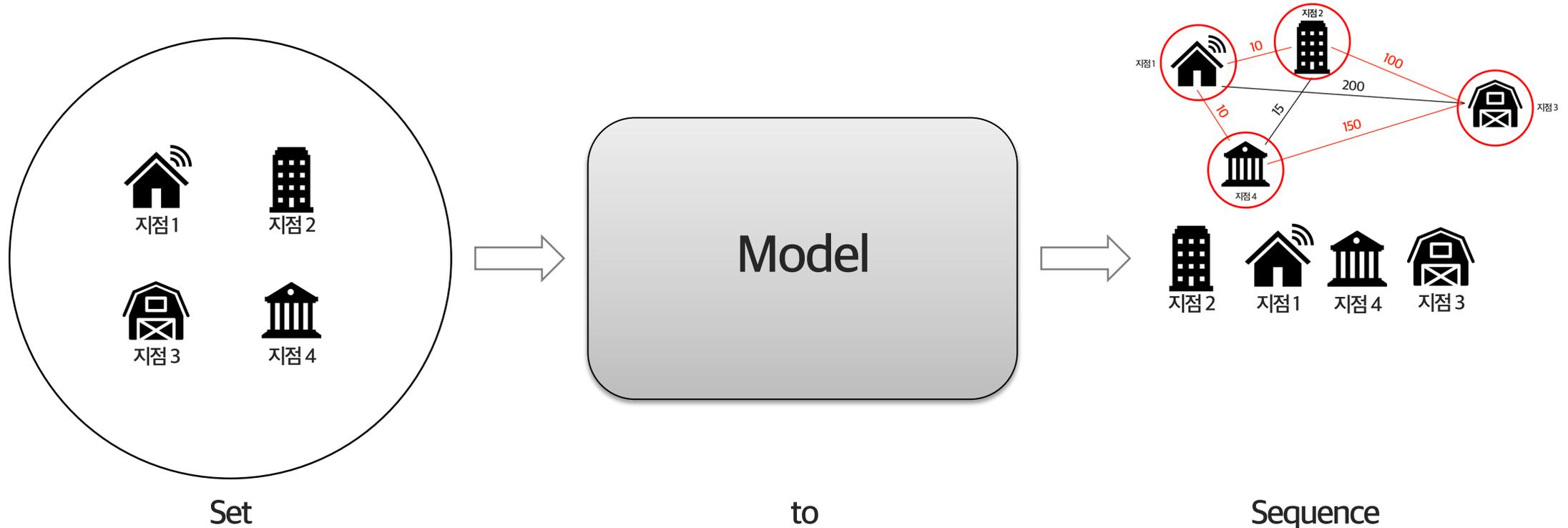
- 방문해야 하는 지점들과 각 지점들 사이의 거리가 주어졌을 때,  
주어진 지점들을 모두 한 번씩만 방문하고 출발 지점으로 돌아오는 최소 길이의 cycle (Optimal Cycle)을 찾는 문제



# TSP is Set2Seq Problem

❖ TSP는 Seq2Seq(Sequence-to-Sequence)과 비슷한 방식으로 모델링 가능

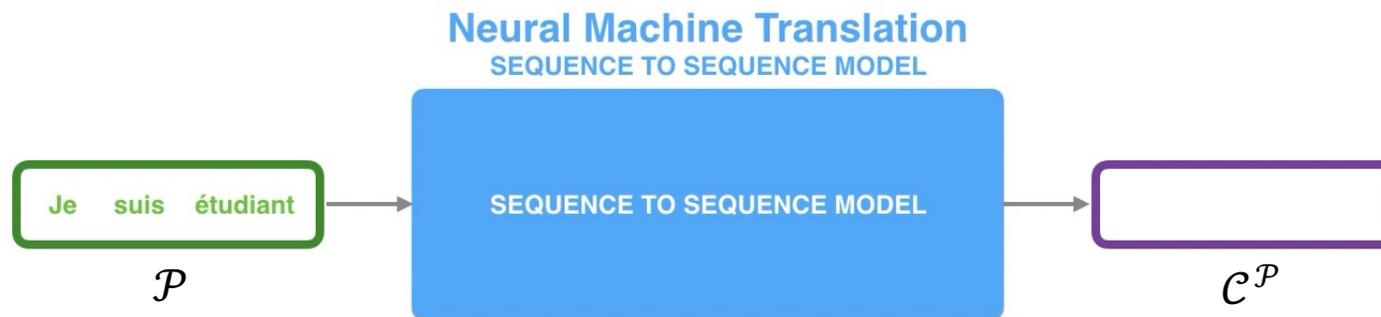
- Seq2Seq 모델은 sequence를 입력받아 sequence를 출력하는 모델
- 결과물인 각 지점의 방문 순서를 sequence라고 한다면, Seq2Seq과 비슷하게 모델링 가능 → Set-to-Sequence( $\approx$  Seq2Seq)



# Seq2Seq Model

## ❖ Seq2Seq(Sequence-to-sequence) model

- 두 개의 sequence로 구성된 training pair  $(\mathcal{P}, \mathcal{C}^{\mathcal{P}})$ 를 학습 데이터로 활용하여, 조건부 확률을 기반으로 sequence  $\mathcal{P}$ 를 입력받으면 sequence  $\mathcal{C}^{\mathcal{P}}$ 를 출력할 수 있도록 학습하는 모델
- Seq2Seq 모델의 학습은 **사전에 정의된 Output Dictionary**를 기반으로 함



### Output Dictionary

a  
am  
are  
I  
is  
professor  
he  
she  
student  
the  
you  
...

# Seq2Seq Model

## ❖ Seq2Seq(Sequence-to-sequence) model

- 두 개의 sequence로 구성된 training pair  $(\mathcal{P}, \mathcal{C}^{\mathcal{P}})$ 를 학습 데이터로 활용하여, 조건부 확률을 기반으로 sequence  $\mathcal{P}$ 를 입력받으면 sequence  $\mathcal{C}^{\mathcal{P}}$ 를 출력하는 모델 학습
- Seq2Seq 모델의 학습은 **사전에 정의된 Output Dictionary**를 기반으로 함



지점 1



지점 2



지점 3



지점 4



SEQUENCE TO SEQUENCE  
MODEL



지점 2



지점 1



지점 4



지점 3

Output Dictionary

지점 1

지점 2

지점 3

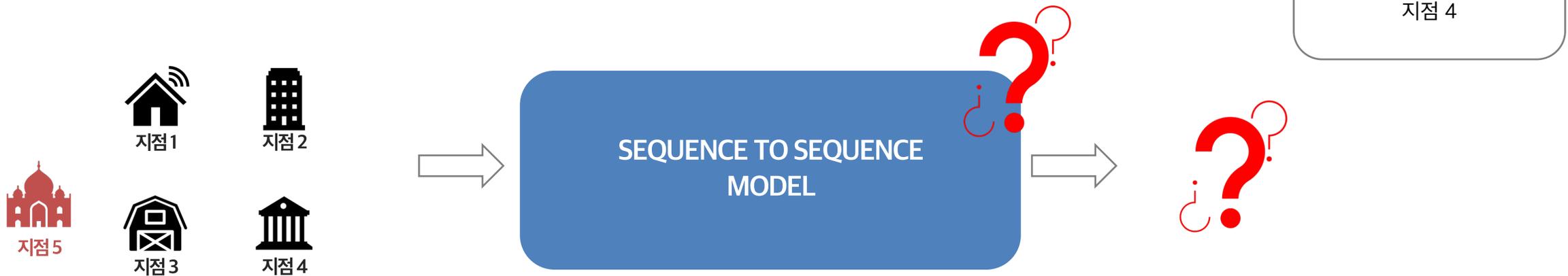
지점 4

# Seq2Seq Model

## ❖ Seq2Seq(Sequence-to-sequence) model

- 두 개의 sequence로 구성된 training pair  $(\mathcal{P}, \mathcal{C}^{\mathcal{P}})$ 를 학습 데이터로 활용하여, 조건부 확률을 기반으로 sequence  $\mathcal{P}$ 를 입력받으면 sequence  $\mathcal{C}^{\mathcal{P}}$ 를 출력하는 모델 학습
- Seq2Seq 모델의 학습은 **사전에 정의된 Output Dictionary**를 기반으로 함

**∴ 입력 지점의 수에 따라 output dictionary가 달라져야 하는 TSP 문제에 바로 적용하는 것은 불가능**



ML Driven Heuristics for Combinatorial Optimization Problems - Core Algorithms

# Pointer Networks

# Pointer Networks [2015 NeurIPS]

## ❖ Pointer Networks

- 2024.05.15 Google Scholar 기준 인용수 3,512회

### Pointer Networks

**Oriol Vinyals\***  
Google Brain

**Meire Fortunato\***  
Department of Mathematics, UC Berkeley

**Navdeep Jaitly**  
Google Brain

#### Abstract

We introduce a new neural architecture to learn the conditional probability of an output sequence with elements that are discrete tokens corresponding to positions in an input sequence. Such problems cannot be trivially addressed by existent approaches such as sequence-to-sequence [1] and Neural Turing Machines [2], because the number of target classes in each step of the output depends on the length of the input, which is variable. Problems such as sorting variable sized sequences, and various combinatorial optimization problems belong to this class. Our model solves the problem of variable size output dictionaries using a recently proposed mechanism of neural attention. It differs from the previous attention attempts in that, instead of using attention to blend hidden units of an encoder to a context vector at each decoder step, it uses attention as a pointer to select a member of the input sequence as the output. We call this architecture a Pointer Net (Ptr-Net). We show Ptr-Nets can be used to learn approximate solutions to three challenging geometric problems – finding planar convex hulls, computing Delaunay triangulations, and the planar Travelling Salesman Problem – using training examples alone. Ptr-Nets not only improve over sequence-to-sequence with input attention, but also allow us to generalize to variable size output dictionaries. We show that the learnt models generalize beyond the maximum lengths they were trained on. We hope our results on these tasks will encourage a broader exploration of neural learning for discrete problems.

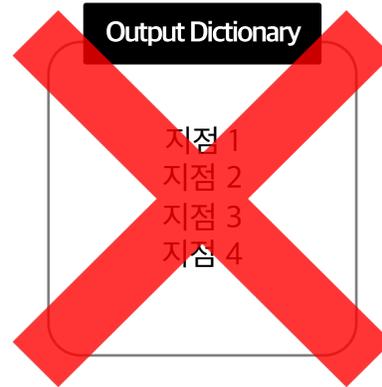
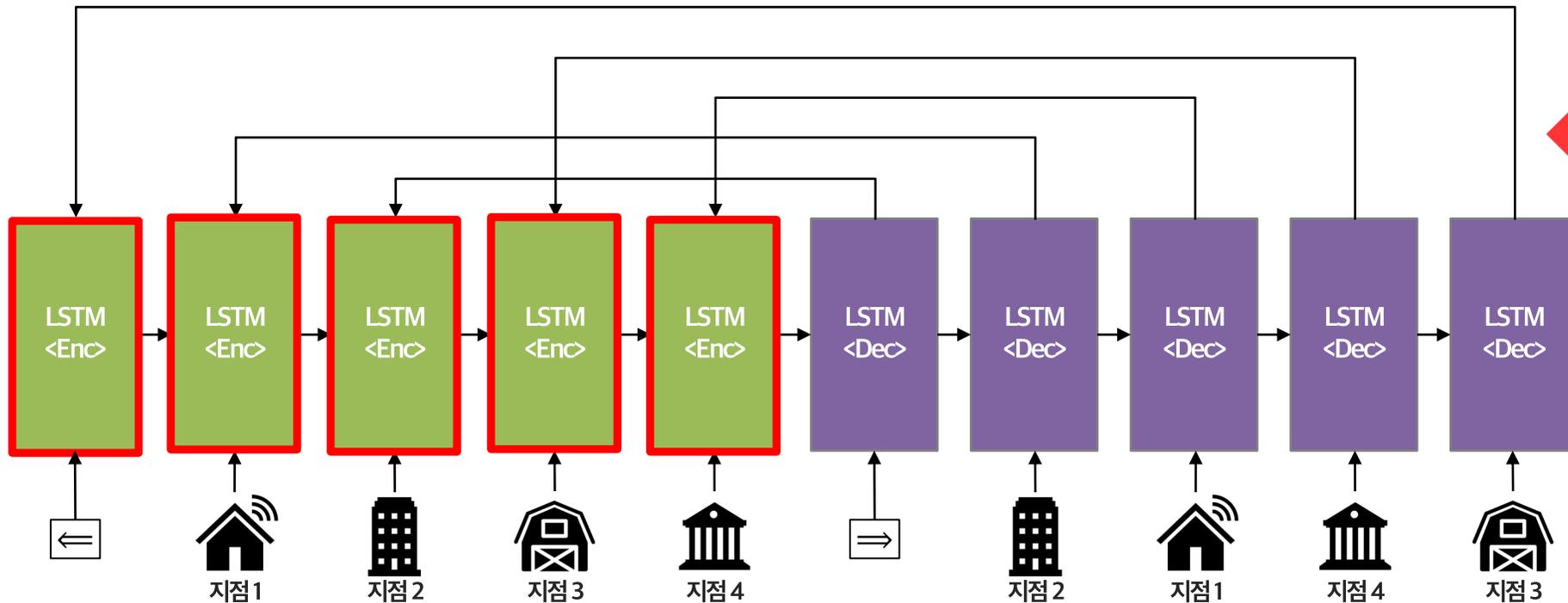
# Pointer Networks [2015 NeurIPS]

## ❖ Ptr-Net

- 입력되는 지점의 수에 따라 달라지는 output dictionary를 반영할 수 없는 기존 Seq2Seq 모델의 한계를 극복하기 위한 Ptr-Net 구조 제안 → Output Dictionary 필요 없음!
- 기존 Attention 구조의 변형을 통하여 입력 Sequence의 요소 중 하나를 '선택(Point)'할 수 있도록 하는 Network



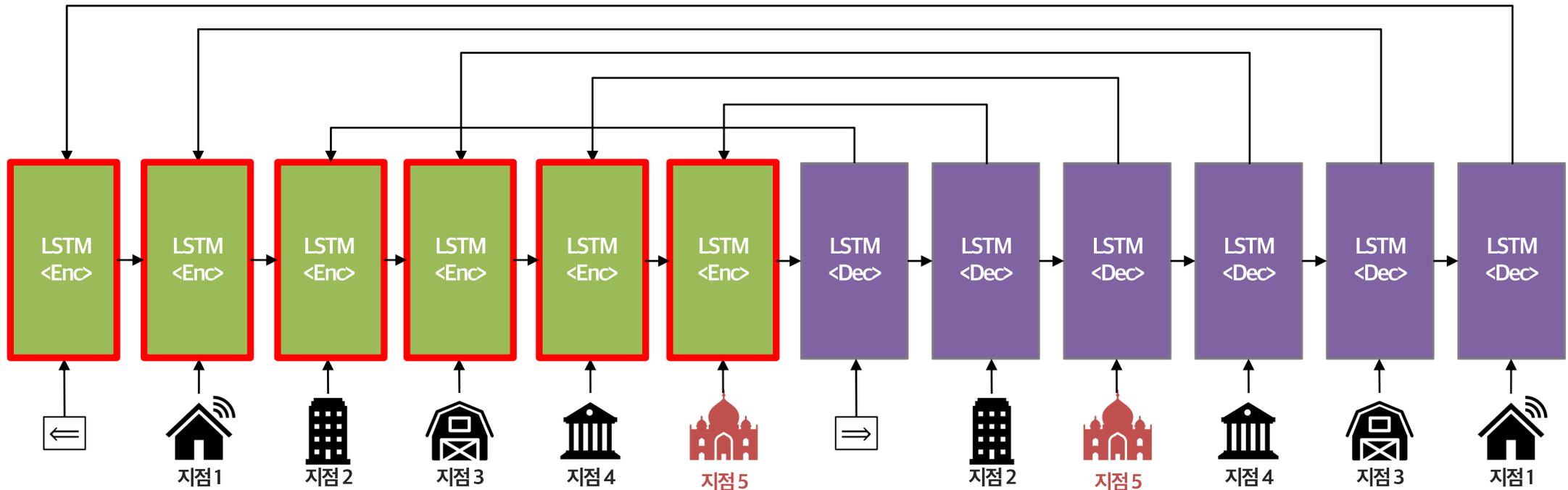
지점 5



# Pointer Networks [2015 NeurIPS]

## ❖ Ptr-Net

- 입력되는 지점의 수에 따라 달라지는 output dictionary를 반영할 수 없는 기존 Seq2Seq 모델의 한계를 극복하기 위한 Ptr-Net 구조 제안 → Output Dictionary 필요 없음!
- 기존 Attention 구조의 변형을 통하여 입력 Sequence의 요소 중 하나를 '선택(Point)'할 수 있도록 하는 Network

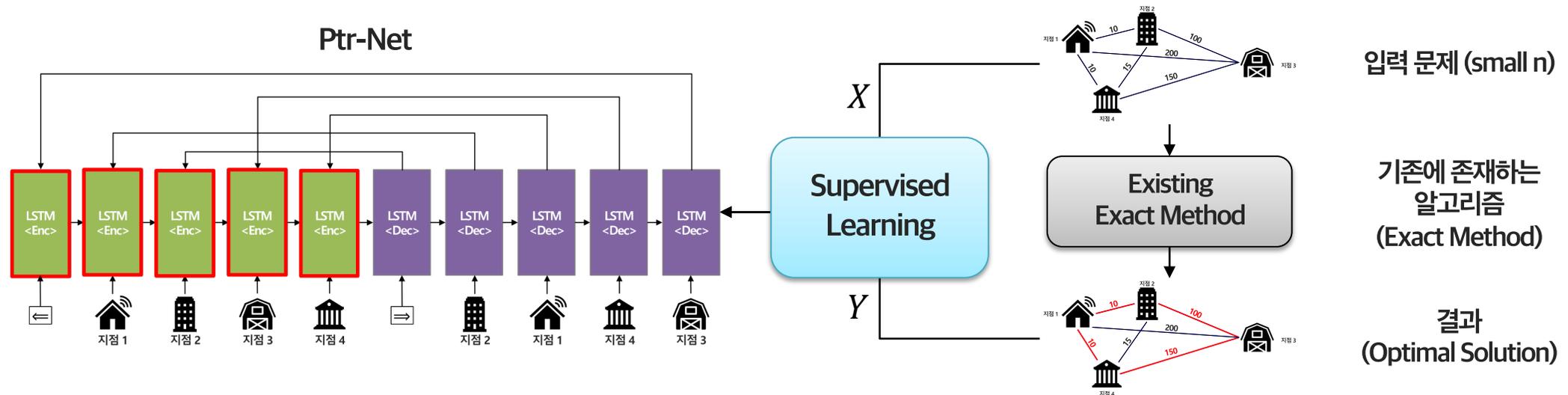


# Pointer Networks [2015 NeurIPS]

학습 방식 상의 한계

## ❖ 본 논문의 학습 방식 : '지도학습'

- 학습 시에 조합최적화 문제를 입력함과 동시에, 해당 문제의 정답인 Optimal Solution을 입력해주어야 함
- Optimal Solution을 구할 수 없는 문제의 경우 학습이 불가능
  - 특히 큰 사이즈의 문제의 경우, Optimal Solution을 구하는 것은 매우 어렵거나 불가능

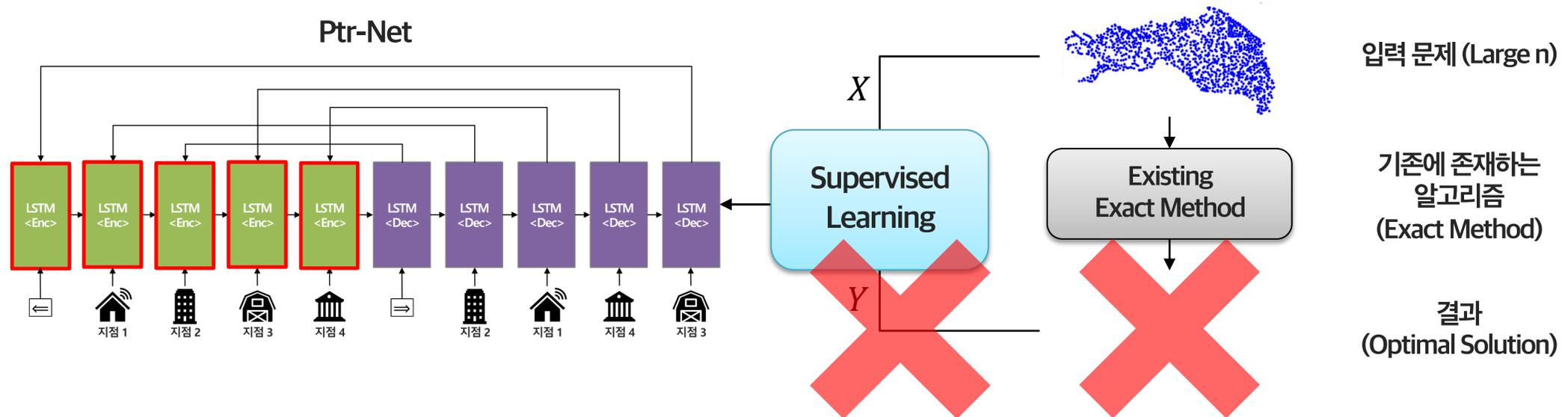


# Pointer Networks [2015 NeurIPS]

학습 방식 상의 한계

## ❖ 본 논문의 학습 방식 : '지도학습'

- 학습 시에 조합최적화 문제를 입력함과 동시에, 해당 문제의 정답인 Optimal Solution을 입력해주어야 함
- Optimal Solution을 구할 수 없는 문제의 경우 학습이 불가능
  - 특히 큰 사이즈의 문제의 경우, Optimal Solution을 구하는 것은 매우 어렵거나 불가능

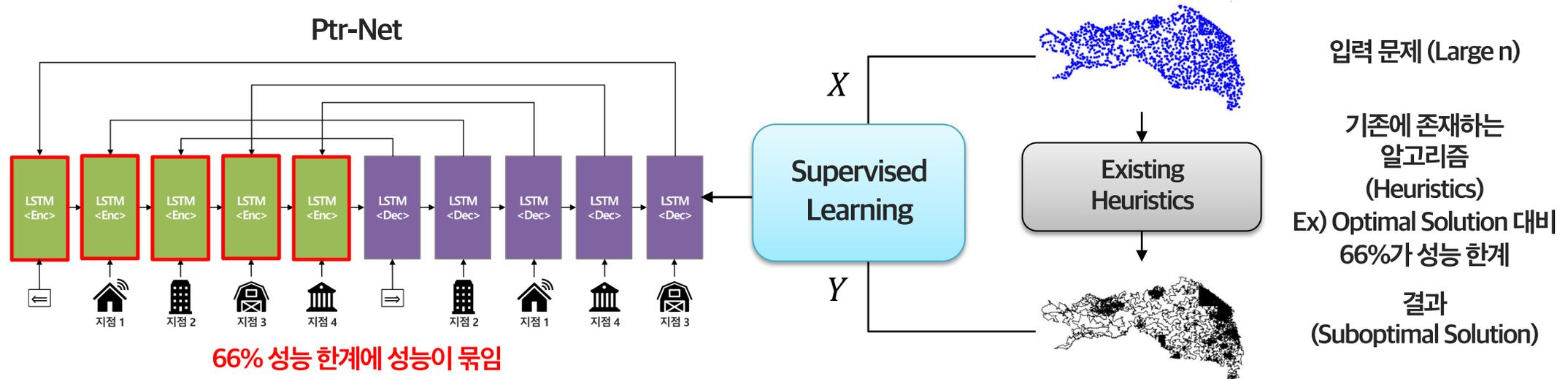


# Pointer Networks [2015 NeurIPS]

학습 방식 상의 한계

## ❖ 본 논문의 학습 방식 : '지도학습'

- Optimal Solution을 구할 수 없는 경우 기존 Heuristic 알고리즘을 활용하여 Suboptimal Solution을 구하여 정답 레이블로 활용한다면 정답으로 주어진 Suboptimal Solution의 Quality에 모델 성능이 영향을 받게 됨



ML Driven Heuristics for Combinatorial Optimization Problems - Core Algorithms

# Neural Combinatorial Optimization with Reinforcement Learning

# Neural Combinatorial Optimization with Reinforcement Learning [2017 ICLR]

## ❖ Neural Combinatorial Optimization with Reinforcement Learning

- 2024.05.15 Google Scholar 기준 인용수 1,702회

### NEURAL COMBINATORIAL OPTIMIZATION WITH REINFORCEMENT LEARNING

**Irwan Bello\*, Hieu Pham\*, Quoc V. Le, Mohammad Norouzi, Samy Bengio**  
Google Brain  
{ibello,hyhieu,qvl,mnorouzi,bengio}@google.com

#### ABSTRACT

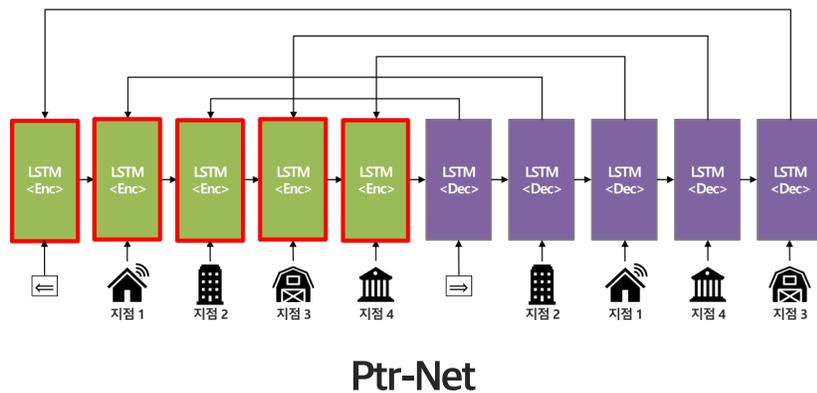
This paper presents a framework to tackle combinatorial optimization problems using neural networks and reinforcement learning. We focus on the traveling salesman problem (TSP) and train a recurrent neural network that, given a set of city coordinates, predicts a distribution over different city permutations. Using negative tour length as the reward signal, we optimize the parameters of the recurrent neural network using a policy gradient method. We compare learning the network parameters on a set of training graphs against learning them on individual test graphs. Despite the computational expense, without much engineering and heuristic designing, Neural Combinatorial Optimization achieves close to optimal results on 2D Euclidean graphs with up to 100 nodes. Applied to the KnapSack, another NP-hard problem, the same method obtains optimal solutions for instances with up to 200 items.

# Neural Combinatorial Optimization with Reinforcement Learning [2017 ICLR]

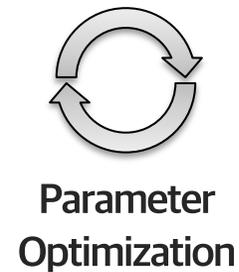
❖ 기존의 Ptr-Net의 파라미터 최적화에 강화학습을 도입할 것을 제안

- 사용하는 모델 구조는 기존 Ptr-Net과 동일
- Ptr-Net의 파라미터 최적화에 강화학습을 도입하여 지도학습 방식으로 인해 발생하는 한계 극복
- Search Strategy 도입

Model



Training Method



+ Search Strategy

# Neural Combinatorial Optimization with Reinforcement Learning [2017 ICLR]

## ❖ Optimization with Policy Gradients - Actor-critic training

- Actor [Ptr-Net,  $p_\theta(\cdot | s_i)$ ] : 주어진 graph  $s_i$ 에 대해서 최단 경로 추정
- Critic [Critic Network,  $b_{\theta_v}(s_i)$ ] : 주어진 graph  $s_i$ 의 최단 경로 길이의 기댓값 추정

---

### Algorithm 1 Actor-critic training

---

```
1: procedure TRAIN(training set  $S$ , number of training steps  $T$ , batch size  $B$ )
2:   Initialize pointer network params  $\theta$ 
3:   Initialize critic network params  $\theta_v$ 
4:   for  $t = 1$  to  $T$  do
5:      $s_i \sim \text{SAMPLEINPUT}(S)$  for  $i \in \{1, \dots, B\}$ 
6:      $\pi_i \sim \text{SAMPLESOLUTION}(p_\theta(\cdot | s_i))$  for  $i \in \{1, \dots, B\}$ 
7:      $b_i \leftarrow b_{\theta_v}(s_i)$  for  $i \in \{1, \dots, B\}$ 
8:      $g_\theta \leftarrow \frac{1}{B} \sum_{i=1}^B (L(\pi_i | s_i) - b_i) \nabla_\theta \log p_\theta(\pi_i | s_i)$ 
9:      $\mathcal{L}_v \leftarrow \frac{1}{B} \sum_{i=1}^B \|b_i - L(\pi_i)\|_2^2$ 
10:     $\theta \leftarrow \text{ADAM}(\theta, g_\theta)$ 
11:     $\theta_v \leftarrow \text{ADAM}(\theta_v, \nabla_{\theta_v} \mathcal{L}_v)$ 
12:  end for
13:  return  $\theta$ 
14: end procedure
```

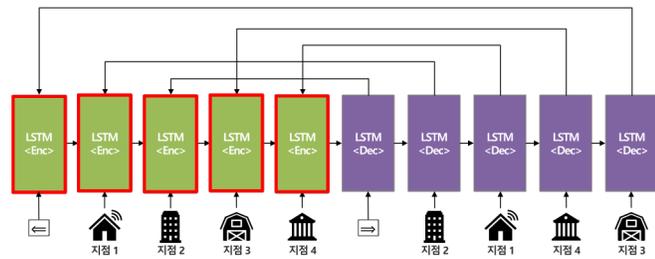
---

$\pi_i$  : Resulting Sequence from Model  
 $L(\pi_i)$  : permutation  $\pi_i$  에 따른 tour length

# Neural Combinatorial Optimization with Reinforcement Learning [2017 ICLR]

## ❖ Optimization with Policy Gradients - Actor-critic training

- Actor [Ptr-Net,  $p_\theta(\cdot | s_i)$ ] : 주어진 graph  $s_i$ 에 대해서 최단 경로 추정
- Critic [Critic Network,  $b_{\theta_v}(s_i)$ ] : 주어진 graph  $s_i$ 의 최단 경로 길이의 기댓값 추정



**Actor**  
Ptr-Net

output :  $\pi_i \sim \text{SampleSolution}(p_\theta(\cdot | s_i))$

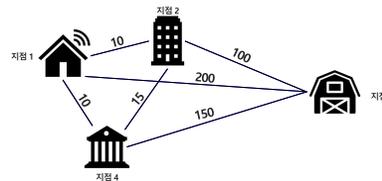
Loss

$$J(\theta | s) = \mathbb{E}_{\pi \sim p_\theta(\cdot | s)} L(\pi | s)$$

Gradient

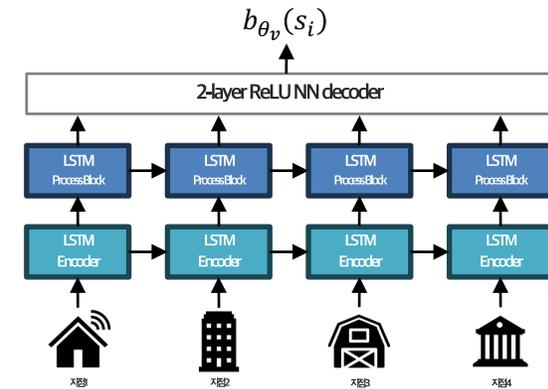
(REINFORCE, Monte-Carlo)

$$g_\theta = \nabla J(\theta | s) \approx \frac{1}{B} \sum_{i=1}^B (L(\pi_i | s_i) - b_i) \nabla_\theta \log p_\theta(\pi_i | s_i)$$



graph instance

$s_i$



**Critic**

LSTM encoder + LSTM process block + 2-layer ReLU NN decoder

output :  $b_i = b_{\theta_v}(s_i)$

MSE Loss

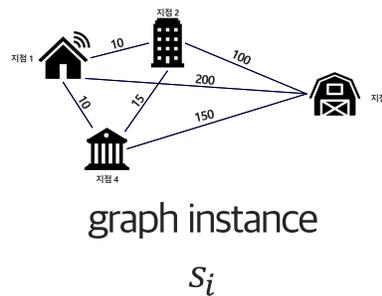
$$L_{\theta_v} = \frac{1}{B} \sum_{i=1}^B \|b_i - L(\pi_i | s_i)\|^2$$

# Neural Combinatorial Optimization with Reinforcement Learning [2017 ICLR]

## ❖ Optimization with Policy Gradients - Actor-critic training

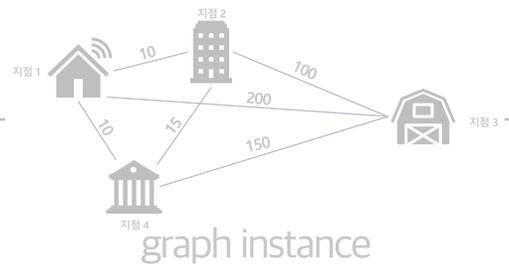
- Actor [Ptr-Net,  $p_{\theta}(\cdot | s_i)$ ]: 주어진 graph  $s_i$ 에 대해서 최단 경로 추정
- Critic [Critic Network,  $b_{\theta_v}(s_i)$ ]: 주어진 graph  $s_i$ 의 최단 경로 길이의 기댓값 추정

Actor  
Ptr-Net



Critic

LSTM encoder + LSTM process block  
+ 2-layer ReLU NN decoder



graph instance  
 $s_i$



# RL Pretraining

output:  $\pi_i \sim \text{Sample}(\text{Solution}(\cdot | s_i))$   
 Actor가 예측한 이 그래프에서의 최적해(방문 순서)

Critic이 예측한 이 그래프의 최적 solution의 전체 cycle 길이

$\pi_i: 1 \rightarrow 3 \rightarrow 4 \rightarrow 2$

Update  
(Gradient Descent)

Update  
(Gradient Descent)

$$\text{Loss: } J(\theta | s) = \mathbb{E}_{\pi \sim p_{\theta}(\cdot | s)} L(\pi | s)$$

Gradient (REINFORCE, Monte-Carlo)

$$g_{\theta} = \frac{1}{B} \sum_{i=1}^B (L(\pi_i | s_i) - b_i) \nabla_{\theta} \log p_{\theta}(\pi_i | s_i)$$

baseline: Gradient의 분산을 줄이는 역할

$b_i: 325$

MSE Loss

$$L_{\theta_v} = \frac{1}{B} \sum_{i=1}^B \|b_i - L(\pi_i | s_i)\|_2^2$$

Gradient

$\pi_i$  : Resulting Sequence from Model  
 $L(\pi_i)$  : permutation  $\pi_i$  에 따른 tour length

# Neural Combinatorial Optimization with Reinforcement Learning [2017 ICLR]

종료

## Actor-Critic Reinforcement Learning Algorithms

고려대학교 박영준

### Actor-Critic Reinforcement Learning Algo

발표자: 박영준

📅 2018년 10월 26일  
 🕒 오후 1시 30분 ~  
 📍 고려대학교 신공학관 218호

👉 [세미나 정보 보기](#)

stochastic training  
 이어서 최단 경로 추정  
 이어서 최단 경로 길이의 기댓값

2015 LECTURE 7  
**Policy Gradient Methods**  
 REINFORCEMENT LEARNING

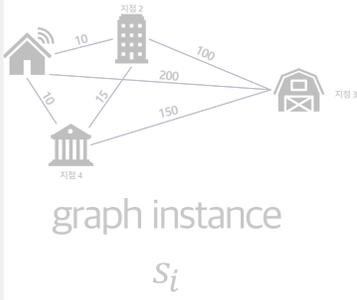
2-layer ReLU NN decoder

### Policy Gradient Algorithms

Date: April 8, 2018 | Estimated Reading Time: 52 min | Author: Lilian Weng

▶ [Table of Contents](#)

- [Updated on 2018-06-30: add two new policy gradient methods, SAC and D4PG.]
- [Updated on 2018-09-30: add a new policy gradient method, TD3.]
- [Updated on 2019-02-09: add SAC with automatically adjusted temperature.]
- [Updated on 2019-06-26: Thanks to Chanseok, we have a version of this post in Korean.]
- [Updated on 2019-09-12: add a new policy gradient method SVPG.]
- [Updated on 2019-12-22: add a new policy gradient method IMPALA.]
- [Updated on 2020-10-15: add a new policy gradient method PPG & some new discussion in PPO.]
- [Updated on 2021-09-19: Thanks to Wenhao & 爱吃猫的鱼, we have this post in Chinese1 & Chinese2].



output :  
 $g_\theta = \frac{1}{B} \sum_{i=1}^B$

REINFORCE Loss

MSE Loss

<https://youtu.be/KHZVXao4qXs?si=6DrHNbQWQfjFLrim>  
<https://lilianweng.github.io/posts/2018-04-08-policy-gradient/>

# Search Strategy

# Neural Combinatorial Optimization with Reinforcement Learning [2017 ICLR]

## ❖ Search Strategy

- Model에서 Inference를 어떻게 할 것인가?
- 본 논문에서는 두 가지 Search Strategy를 고려함

### Sampling

Greedy Decoding 대신,

- 한 Graph Instance에 대해
- 파라미터가 고정된 한 모델로부터
- Stochastic하게 여러 Solution을 도출하고
- 전체 길이가 가장 짧은 Solution을
- 모델의 Inference로 처리하자!

### Active Search

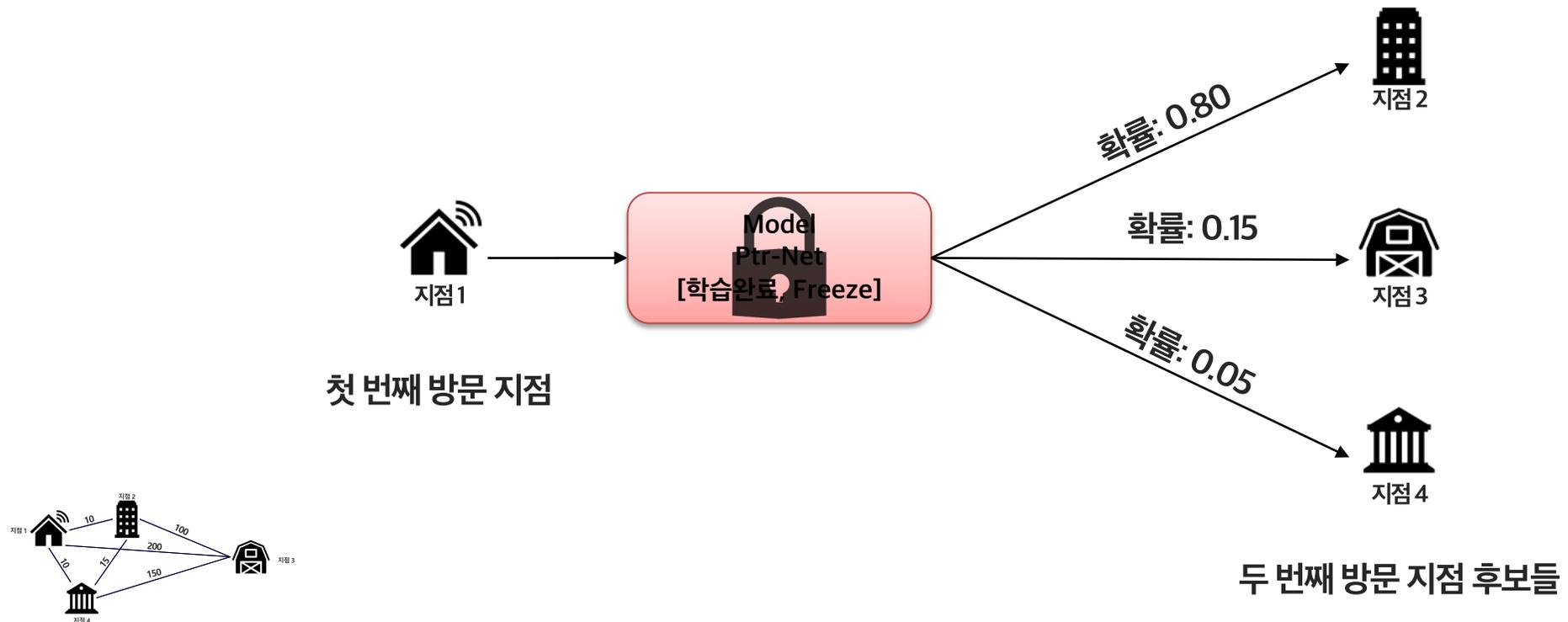
- 한 Graph Instance에 대해
- 한 모델로부터 여러 Solution을 Sampling하는 과정에서
- Sampling할 때마다 모델의 파라미터를 업데이트하며
- Inference하자!

# Neural Combinatorial Optimization with Reinforcement Learning [2017 ICLR]

## ❖ No Search Strategy - Greedy Decoding

- Stochasticity 없이, 확률 값이 가장 높은 것으로 Inference

\*모델이 첫 번째 지점을 입력받고, 두 번째 지점을 골라야 하는 상황 (Inference 과정의 일부)



Sampling

Greedy Decoding 대신,

- 한 Graph Instance에 대해
- 파라미터가 고정된 한 모델로부터
- Stochastic하게 여러 Solution을 도출하고
- 전체 길이가 가장 짧은 Solution을
- 모델의 Inference로 처리하자!

Active Search

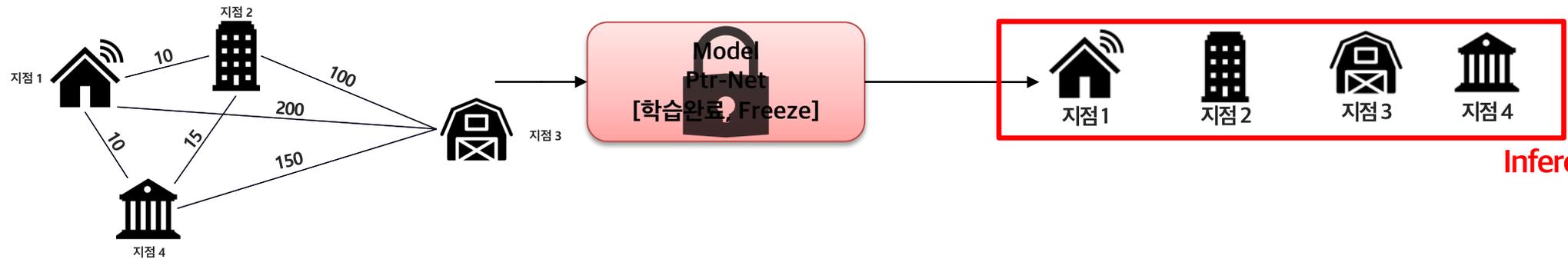
- 한 Graph Instance에 대해
- 한 모델로부터 여러 Solution을 Sampling하는 과정에서
- Sampling할 때마다 모델의 파라미터를 업데이트하며
- Inference하자!

# Neural Combinatorial Optimization with Reinforcement Learning [2017 ICLR]

## ❖ No Search Strategy - Greedy Decoding

- Stochasticity 없이, 확률 값이 가장 높은 것으로 Inference

\*모델이 첫 번째 지점을 입력받고, 두 번째 지점을 골라야 하는 상황 (Inference 과정의 일부)



Inference Result

Sampling

Greedy Decoding 대신,

- 한 Graph Instance에 대해
- 파라미터가 고정된 한 모델로부터
- Stochastic하게 여러 Solution을 도출하고
- 전체 길이가 가장 짧은 Solution을
- 모델의 Inference로 처리하자!

Active Search

- 한 Graph Instance에 대해
- 한 모델로부터 여러 Solution을 Sampling하는 과정에서
- Sampling할 때마다 모델의 파라미터를 업데이트하며
- Inference하자!

# Neural Combinatorial Optimization with Reinforcement Learning [2017 ICLR]

**Sampling**

Greedy Decoding 대신,

- 한 Graph Instance에 대해
- 파라미터가 고정된 한 모델로부터
- Stochastic하게 여러 Solution을 도출하고
- 전체 길이가 가장 짧은 Solution을
- 모델의 Inference로 처리하자!

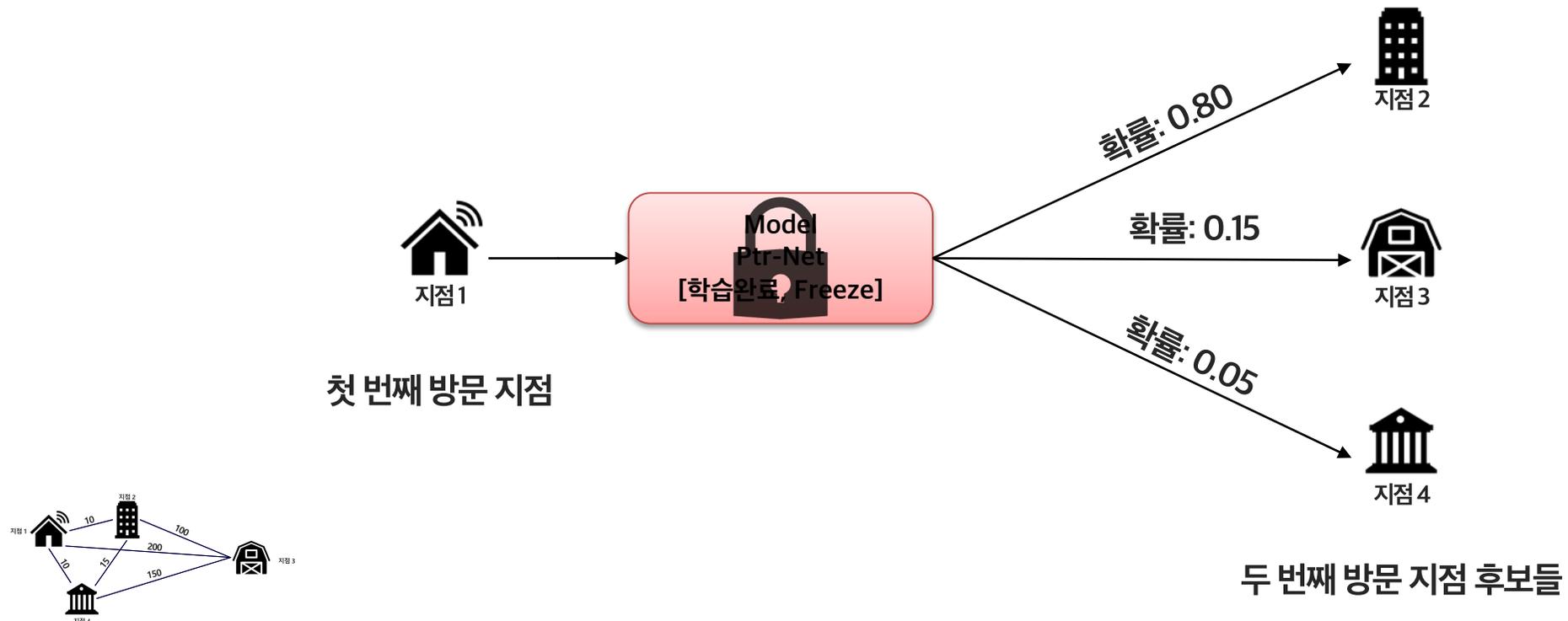
**Active Search**

- 한 Graph Instance에 대해
- 한 모델로부터 여러 Solution을 Sampling하는 과정에서
- Sampling할 때마다 모델의 파라미터를 업데이트하며
- Inference하자!

## ❖ Search Strategy - Sampling

- 비록 확률이 낮은 선택지라도 선택될 수 있도록 하고, 여러 번의 샘플링 후 전체 길이가 가장 짧은 것을 Inference로 처리

\*모델이 첫 번째 지점을 입력받고, 두 번째 지점을 골라야 하는 상황 (Inference 과정의 일부)



# Neural Combinatorial Optimization with Reinforcement Learning [2017 ICLR]

**Sampling**

Greedy Decoding 대신,  
 • 한 Graph Instance에 대해  
 • 파라미터가 고정된 한 모델로부터  
 • Stochastic하게 여러 Solution을 도출하고  
 • 전체 길이가 가장 짧은 Solution을  
 • 모델의 Inference로 처리하자!

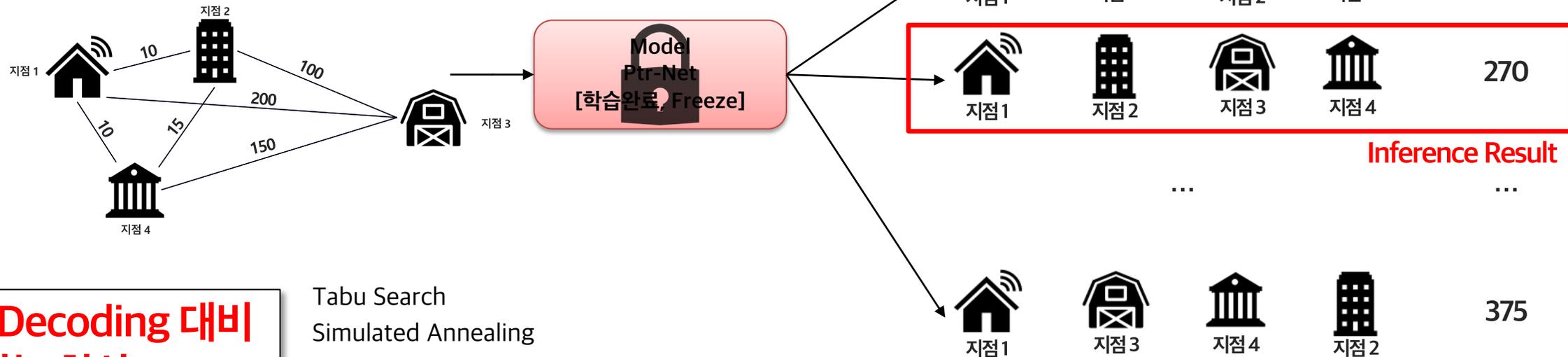
**Active Search**

• 한 Graph Instance에 대해  
 • 한 모델로부터 여러 Solution을 Sampling하는 과정에서  
 • Sampling할 때마다 모델의 파라미터를 업데이트하며  
 • Inference하자!

## ❖ Search Strategy - Sampling

- 비록 확률이 낮은 선택지라도 선택될 수 있도록 하고, 여러 번의 샘플링 후 전체 길이가 가장 짧은 것을 Inference로 처리

\*모델이 첫 번째 지점을 입력받고, 두 번째 지점을 골라야 하는 상황 (Inference 과정의 일부)



**Greedy Decoding 대비 상당한 성능 향상**

- Tabu Search
- Simulated Annealing
- Mutation in Genetic Algorithm
- [DL Optimizer] Momentum

# Neural Combinatorial Optimization with Reinforcement Learning [2017 ICLR]

## ❖ Search Strategy - Active Search

- Sampling을 진행하면서도 모델의 파라미터를 업데이트하는 방법

**Sampling**

Greedy Decoding 대신,

- 한 Graph Instance에 대해
- 파라미터가 고정된 한 모델로부터
- Stochastic하게 여러 Solution을 도출하고
- 전체 길이가 가장 짧은 Solution을
- 모델의 Inference로 처리하자!

**Active Search**

- 한 Graph Instance에 대해
- 한 모델로부터 여러 Solution을 Sampling하는 과정에서
- Sampling할 때마다 모델의 파라미터를 업데이트하며
- Inference하자!

**Sampling**

Greedy Decoding 대신,

- 한 Graph Instance에 대해
- 파라미터가 고정된 한 모델로부터
- Stochastic하게 여러 Solution을 도출하고
- 전체 길이가 가장 짧은 Solution을
- 모델의 Inference로 처리하자!

**Active Search**

- 한 Graph Instance에 대해
- 한 모델로부터 여러 Solution을 Sampling하는 과정에서
- Sampling할 때마다 모델의 파라미터를 업데이트하며
- Inference하자!

# Neural Combinatorial Optimization with Reinforcement Learning [2017 ICLR]

## ❖ Search Strategy - Active Search

- Sampling을 진행하면서도 모델의 파라미터를 업데이트하는 방법

Sampling

Greedy Decoding 대신,

- 한 Graph Instance에 대해
- 파라미터가 고정된 한 모델로부터
- Stochastic하게 여러 Solution을 도출하고
- 전체 길이가 가장 짧은 Solution을
- 모델의 Inference로 처리하자!

Active Search

- 한 Graph Instance에 대해
- 한 모델로부터 여러 Solution을 Sampling하는 과정에서
- Sampling할 때마다 모델의 파라미터를 업데이트하며 Inference하자!

### Algorithm 2 Active Search

```

1: procedure ACTIVESEARCH(input  $s, \theta$ , number of candidates  $K, B, \alpha$ )
2:    $\pi \leftarrow \text{RANDOM SOLUTION}()$ 
3:    $L_\pi \leftarrow L(\pi | s)$ 
4:    $n \leftarrow \lceil \frac{K}{B} \rceil$ 
5:   for  $t = 1 \dots n$  do
6:      $\pi_i \sim \text{SAMPLE SOLUTION}(p_\theta(\cdot | s))$  for  $i \in \{1, \dots, B\}$ 
7:      $j \leftarrow \text{ARGMIN}(L(\pi_1 | s) \dots L(\pi_B | s))$ 
8:      $L_j \leftarrow L(\pi_j | s)$ 
9:     if  $L_j < L_\pi$  then
10:       $\pi \leftarrow \pi_j$ 
11:       $L_\pi \leftarrow L_j$ 
12:     end if
13:      $g_\theta \leftarrow \frac{1}{B} \sum_{i=1}^B (L(\pi_i | s) - b) \nabla_\theta \log p_\theta(\pi_i | s)$ 
14:      $\theta \leftarrow \text{ADAM}(\theta, g_\theta)$ 
15:      $b \leftarrow \alpha \times b + (1 - \alpha) \times (\frac{1}{B} \sum_{i=1}^B b_i)$ 
16:   end for
17:   return  $\pi$ 
18: end procedure

```

RL Pretraining 과정 없이도  
관찰은 Solution 도출 가능

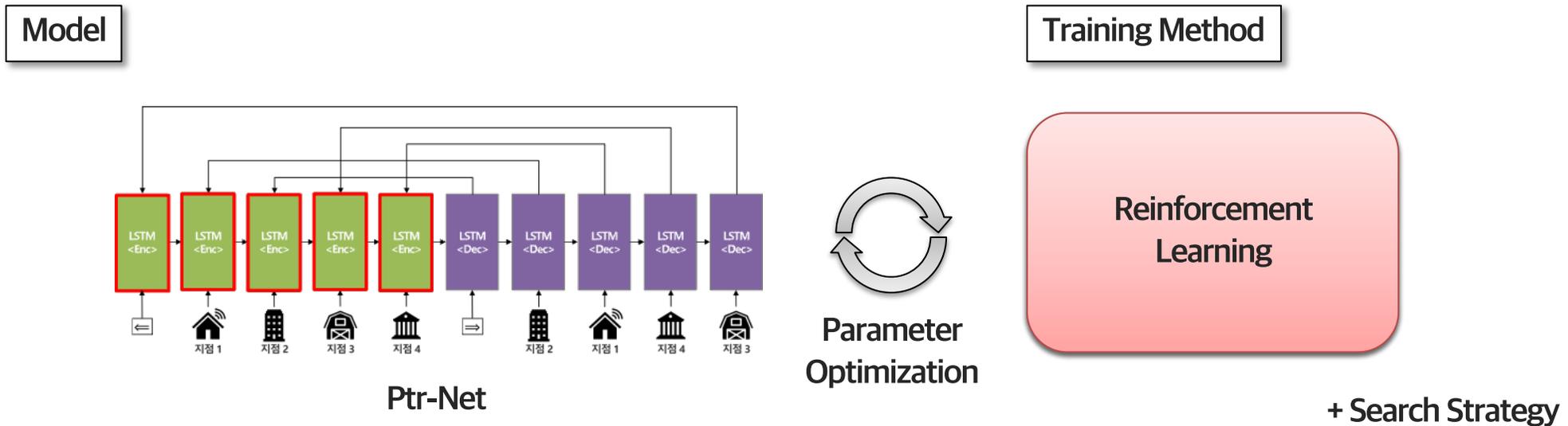
Active Search

- 한 Graph Instance에 대해
- 한 모델로부터 여러 Solution을 Sampling하는 과정에서
- Sampling할 때마다 모델의 파라미터를 업데이트하며
- Inference하자!

# Neural Combinatorial Optimization with Reinforcement Learning [2017 ICLR]

## ❖ 본 논문의 의의

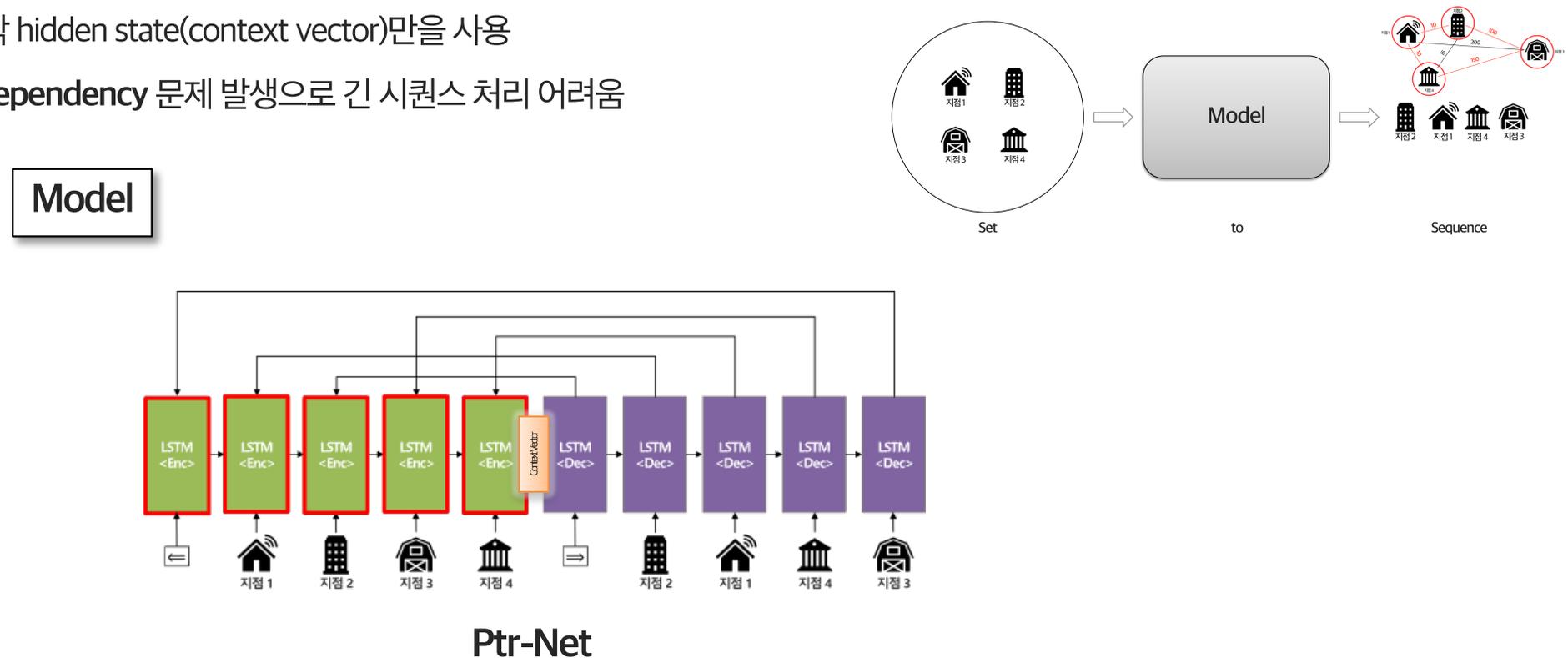
- 모델 학습에 강화학습을 활용하여 지도학습의 한계 극복
- Search Strategy에 대한 연구
- 기존 지도학습 기반 방법론 또는 기존 heuristic 대비 성능 향상



# Neural Combinatorial Optimization with Reinforcement Learning [2017 ICLR]

## ❖ 모델 구조 상의 한계 - Seq2Seq

- 정보의 순차 처리 : 방문 지점 입력 순서에 따라 decoding 결과, 그리고 성능에 차이 발생
  - TSP 문제의 경우, 방문 지점의 입력 순서와 관계 없이 동일한 decoding 결과를 얻을 수 있어야 함(Set2Seq)
- Encoder의 마지막 hidden state(context vector)만을 사용
  - Long-term dependency 문제 발생으로 긴 시퀀스 처리 어려움



ML Driven Heuristics for Combinatorial Optimization Problems - Core Algorithms

# Attention, Learn to Solve Routing Problems!

# Attention, Learn to Solve Routing Problems! [2019 ICLR]

## ❖ Attention, Learn to Solve Routing Problems!

- 2024.05.15 Google Scholar 기준 인용수 1,265회

### ATTENTION, LEARN TO SOLVE ROUTING PROBLEMS!

**Wouter Kool**  
University of Amsterdam  
ORTEC  
w.w.m.kool@uva.nl

**Herke van Hoof**  
University of Amsterdam  
h.c.vanhoof@uva.nl

**Max Welling**  
University of Amsterdam  
CIFAR  
m.welling@uva.nl

#### ABSTRACT

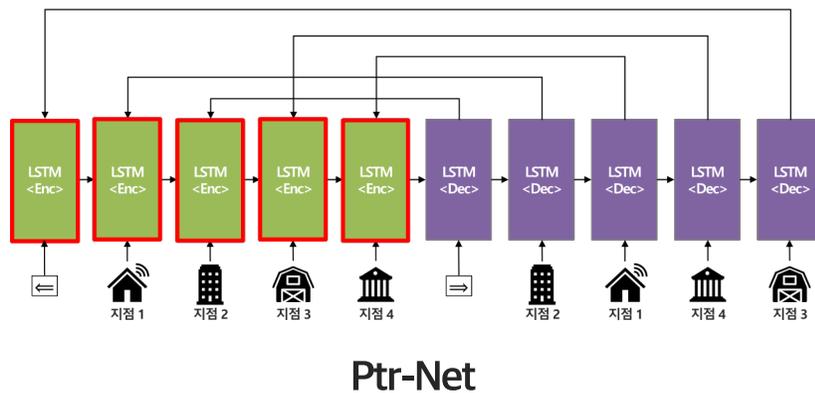
The recently presented idea to learn heuristics for combinatorial optimization problems is promising as it can save costly development. However, to push this idea towards practical implementation, we need better models and better ways of training. We contribute in both directions: we propose a model based on attention layers with benefits over the Pointer Network and we show how to train this model using REINFORCE with a simple baseline based on a deterministic greedy rollout, which we find is more efficient than using a value function. We significantly improve over recent learned heuristics for the Travelling Salesman Problem (TSP), getting close to optimal results for problems up to 100 nodes. With the same hyperparameters, we learn strong heuristics for two variants of the Vehicle Routing Problem (VRP), the Orienteering Problem (OP) and (a stochastic variant of) the Prize Collecting TSP (PCTSP), outperforming a wide range of baselines and getting results close to highly optimized and specialized algorithms.

# Attention, Learn to Solve Routing Problems! [2019 ICLR]

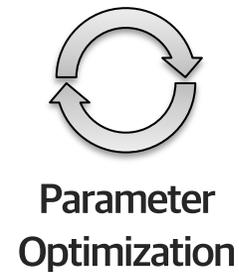
## ❖ Attention을 기반으로 하는 모델 구조를 제안

- 전체적인 구조는 동일 : 문제를 표현하는 그래프를 처리하기 위한 모델 + 강화학습을 이용한 training method
- **Graph Attention**을 기반으로 하는 인코더와 디코더 구조 제안

Model



Training Method

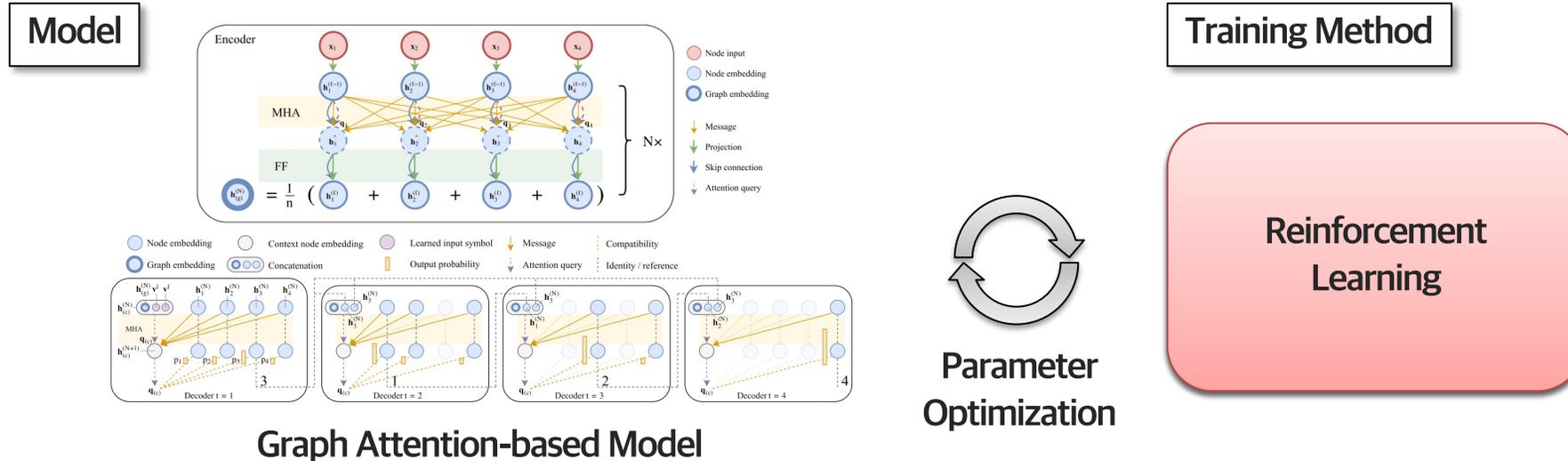


Reinforcement Learning

# Attention, Learn to Solve Routing Problems! [2019 ICLR]

## ❖ Attention을 기반으로 하는 모델 구조를 제안

- 전체적인 구조는 동일 : 문제를 표현하는 그래프를 처리하기 위한 모델 + 강화학습을 이용한 training method
- **Graph Attention**을 기반으로 하는 인코더와 디코더 구조 제안

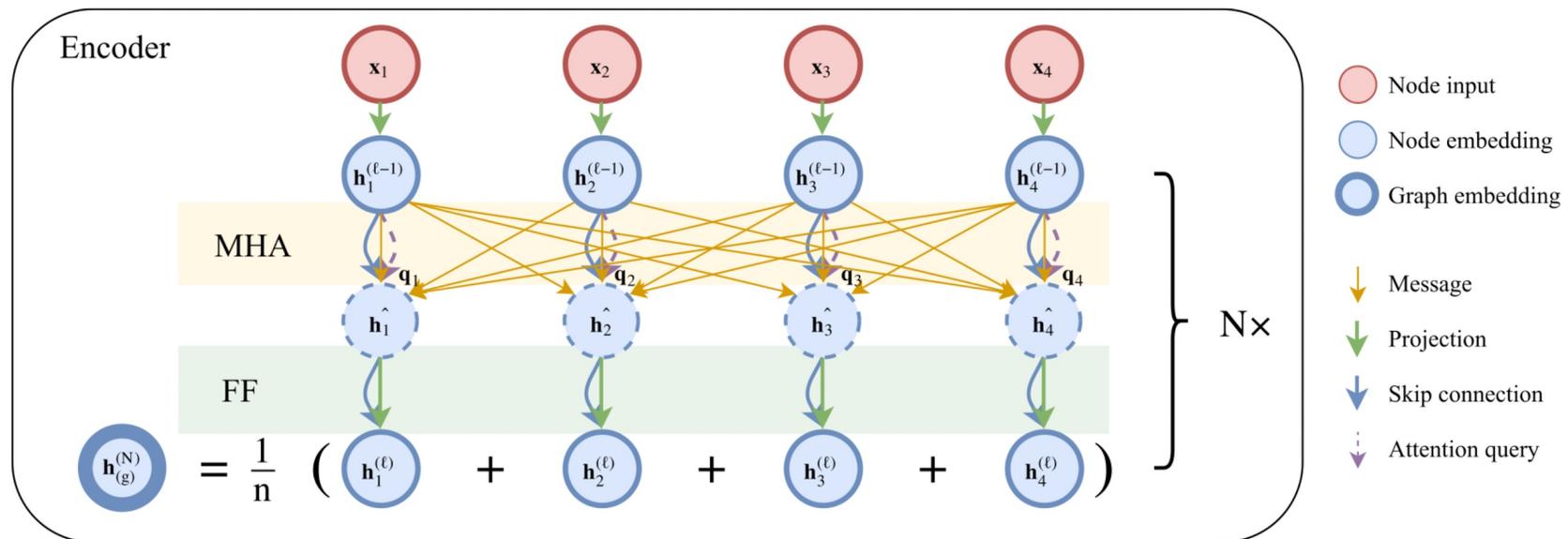


# Attention, Learn to Solve Routing Problems! [2019 ICLR]

## ❖ Model - Encoder

- Graph Attention 기반 인코더 구조 제안
- Transformer architecture의 Multi-head Attention과 같으나, positional encoding을 사용하지 않음

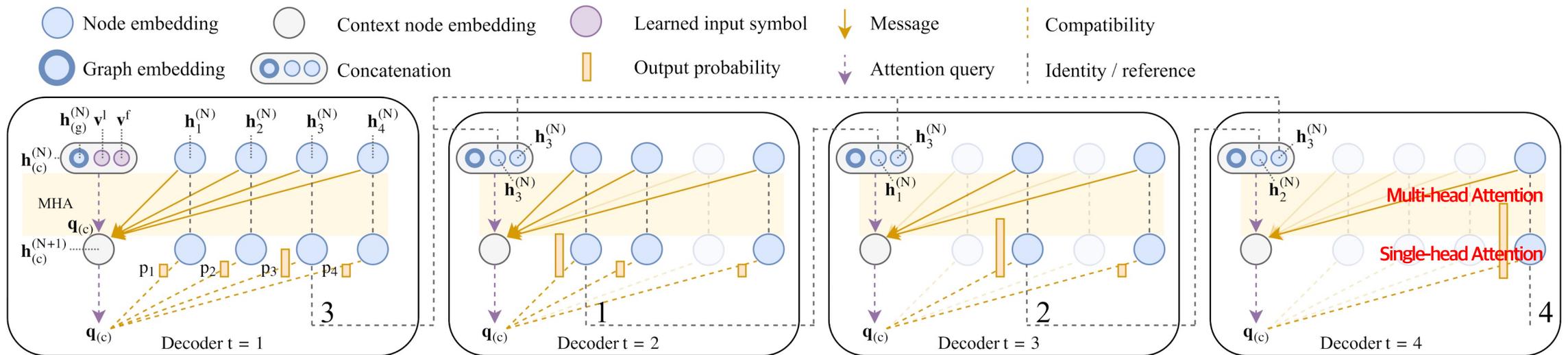
→ 각 방문 지점의 입력 순서와 관계 없이 동일한 node embedding( $\mathbf{h}_i^{(N)}$ ), graph embedding( $\mathbf{h}_{(g)}^{(N)}$ ) 획득 가능



# Attention, Learn to Solve Routing Problems! [2019 ICLR]

## ❖ Model - Decoder

- Decoding context를 보존하기 위하여 context node( $\mathbf{h}_{(c)}^{(N)}$ ) 활용 (graph embedding  $\oplus$  이전 방문 node embedding  $\oplus$  최초 방문 node embedding)
- Node embedding과 context node간의 Multi-head Attention 계산 후,  
Single-head Attention 계산을 통해 최종 확률을 계산하여 다음 방문 지점 결정
- Multi-head Attention 및 Single-head Attention 계산 시 기존에 방문한 지점에 대한 node에는 masking 수행

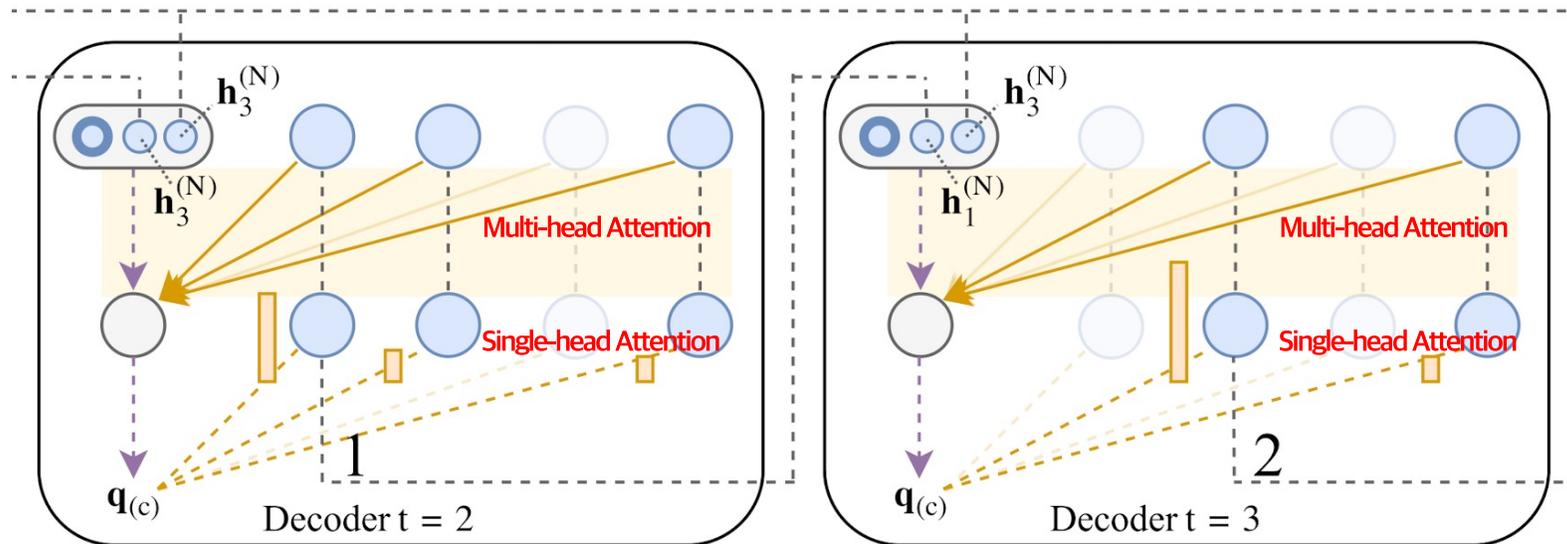


Attention based Decoder

# Attention, Learn to Solve Routing Problems! [2019 ICLR]

## ❖ Model - Decoder

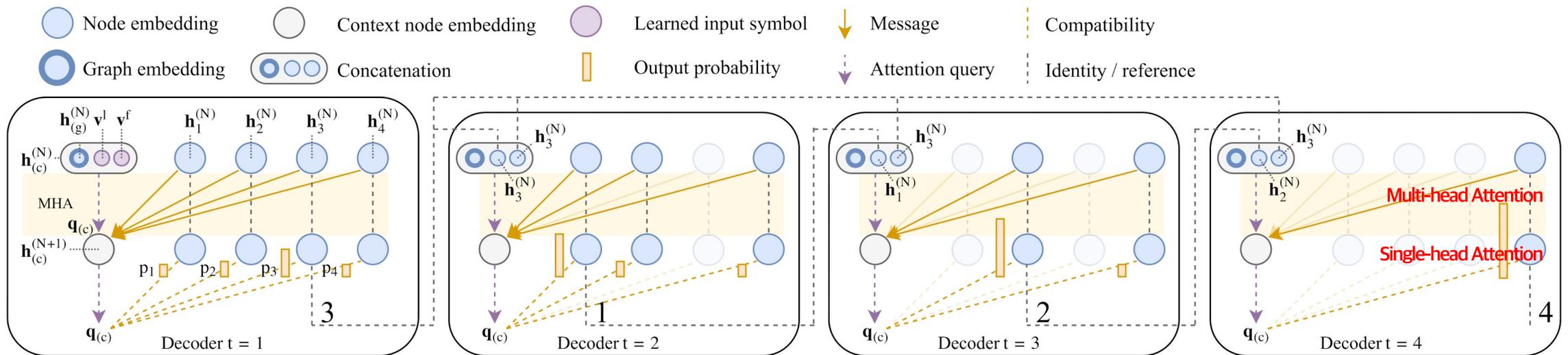
- Decoding context를 보존하기 위하여 context node( $\mathbf{h}_{(c)}^{(N)}$ ) 활용 (graph embedding  $\oplus$  이전 방문 node embedding  $\oplus$  최초 방문 node embedding)
- Node embedding과 context node간의 Multi-head Attention 계산 후,  
Single-head Attention 계산을 통해 최종 확률을 계산하여 다음 방문 지점 결정
- Multi-head Attention 및 Single-head Attention 계산 시 기존에 방문한 지점에 대한 node에는 masking 수행



# Attention, Learn to Solve Routing Problems! [2019 ICLR]

## ❖ Model - Decoder

- Decoding context를 보존하기 위하여 context node( $\mathbf{h}_{(c)}^{(N)}$ ) 활용 (graph embedding  $\oplus$  이전 방문 node embedding  $\oplus$  최초 방문 node embedding)
- Node embedding과 context node간의 Multi-head Attention 계산 후,  
Single-head Attention 계산을 통해 최종 확률을 계산하여 다음 방문 지점 결정
- Multi-head Attention 및 Single-head Attention 계산 시 기존에 방문한 지점에 대한 node에는 masking 수행

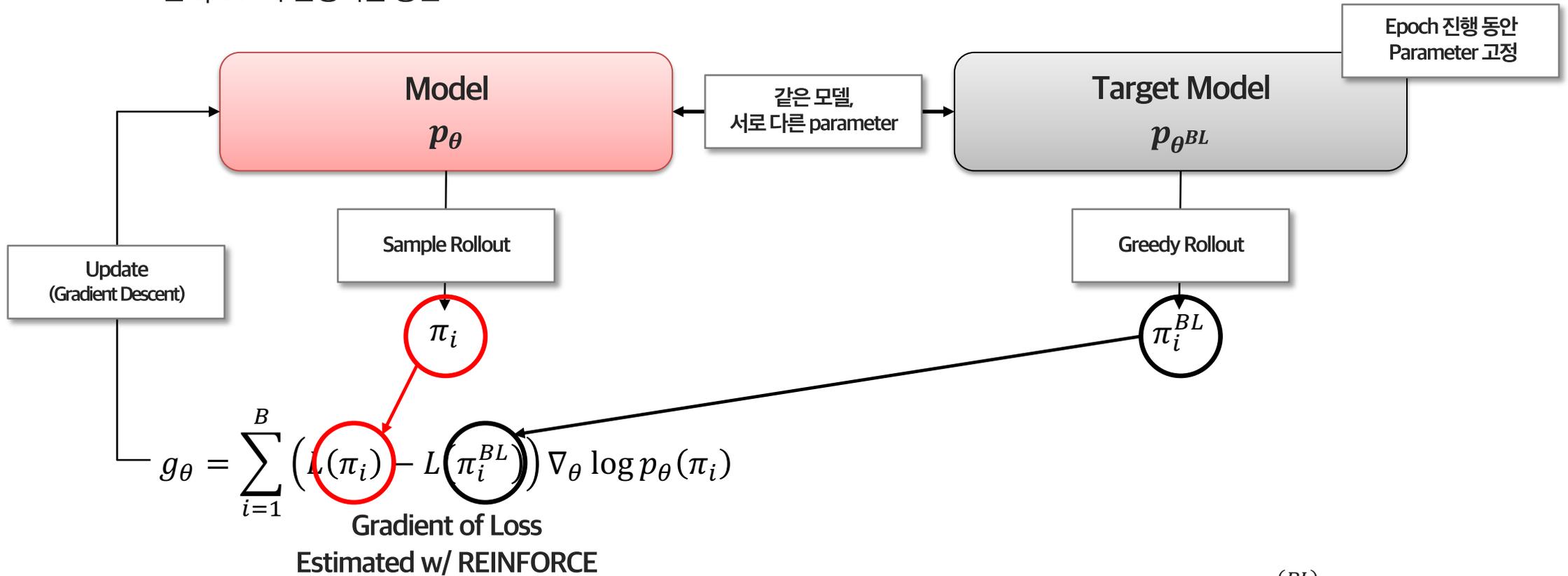


Attention based Decoder

# Attention, Learn to Solve Routing Problems! [2019 ICLR]

## ❖ Optimization with Policy Gradient - REINFORCE with Greedy Rollout Baseline

- 한 epoch이 진행되는 동안

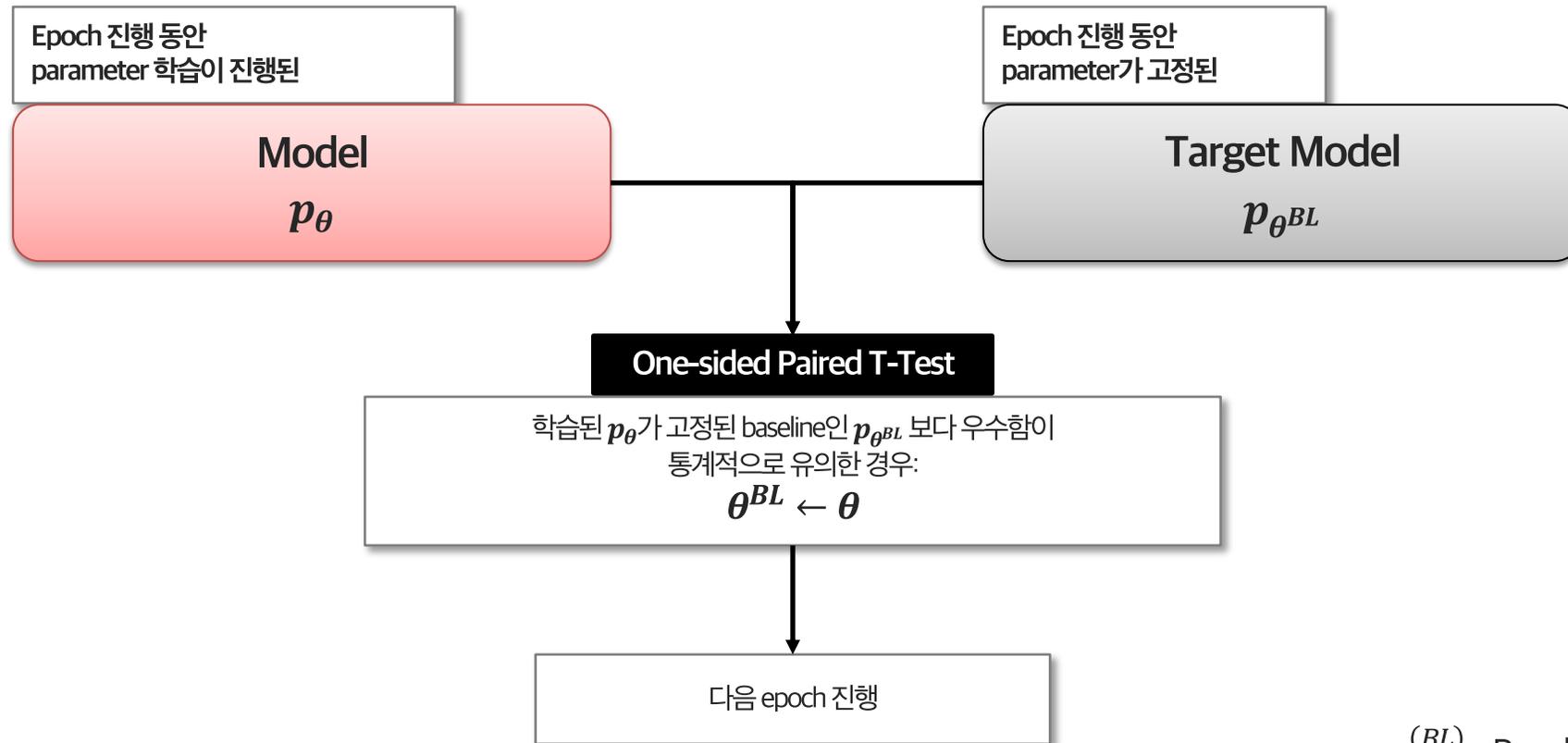


$\pi_i^{(BL)}$  : Resulting Sequence from Model  
 $L(\pi_i^{(BL)})$  : permutation  $\pi_i^{(BL)}$ 에 따른 tour length

# Attention, Learn to Solve Routing Problems! [2019 ICLR]

## ❖ Optimization with Policy Gradient - REINFORCE with Greedy Rollout Baseline

- 한 epoch의 마지막에서

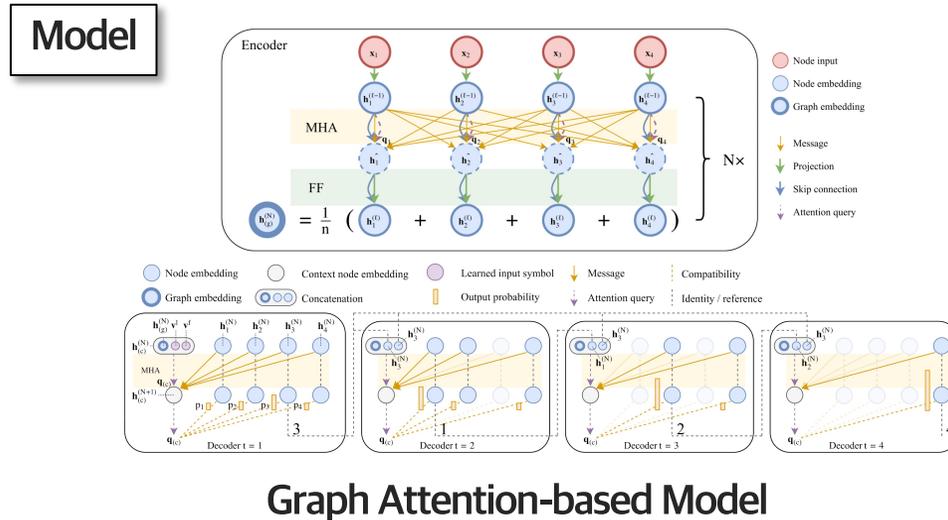


$\pi_i^{(BL)}$  : Resulting Sequence from Model  
 $L(\pi_i^{(BL)})$  : permutation  $\pi_i^{(BL)}$ 에 따른 tour length

# Attention, Learn to Solve Routing Problems! [2019 ICLR]

## ❖ Contribution

- 기존 방법론들과 달리, **입력 순서에 상관 없이 동일한 node embedding 및 결과 획득 가능** → Learning Efficiency
  - 엄밀히는 Set2Seq인 TSP 문제의 본질에 보다 적합함
- **병렬 처리가 보다 용이** → Computational Efficiency
- 기존 Seq2Seq 기반 방법론들 대비 **scalability potential 향상**



## Training Method



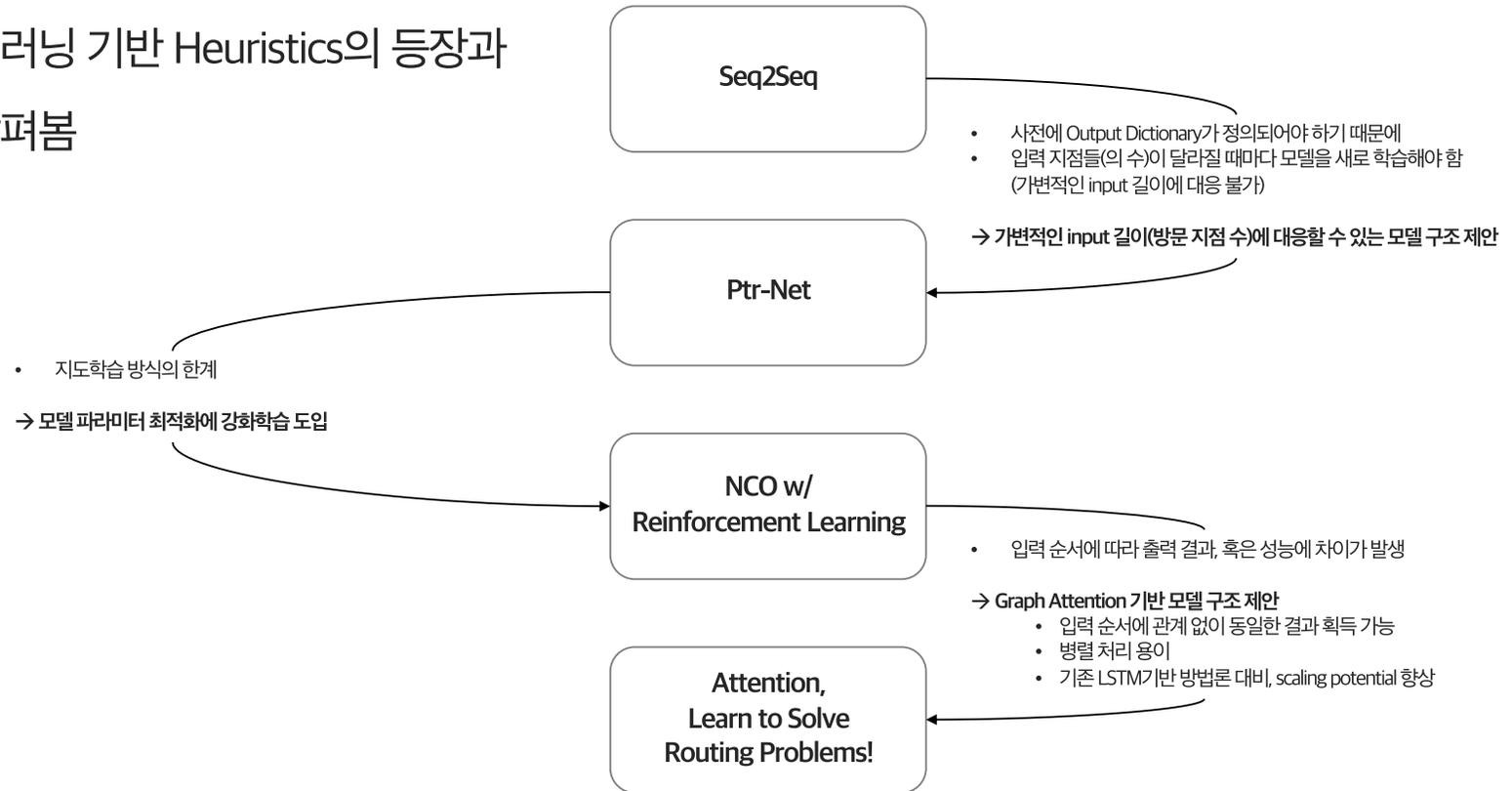
ML Driven Heuristics for Combinatorial Optimization Problems - Core Algorithms

# Conclusion

# Conclusion

ML Driven Heuristics for Combinatorial Optimization Problems - Core Algorithms

- ❖ 최적화 문제와 조합최적화 문제에 대한 Introduction
- ❖ [2019 ICLR] Attention, Learn to Solve Routing Problems! 의 등장까지  
조합최적화 문제를 해결하기 위한 각 머신러닝 기반 Heuristics의 등장과  
이전 방법론의 한계를 극복하는 과정을 살펴봄



# Thank You

# References

ML Driven Heuristics for Combinatorial Optimization Problems - Core Algorithms

1. Mazyavkina, N., Sviridov, S., Ivanov, S., & Burnaev, E. (2021). Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 134, 105400.
2. Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer networks. *Advances in neural information processing systems*, 28.
3. Bello, I., Pham, H., Le, Q. V., Norouzi, M., & Bengio, S. (2016). Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*.
4. Kool, W., Van Hoof, H., & Welling, M. (2018). Attention, learn to solve routing problems!. *arXiv preprint arXiv:1803.08475*.