Neural Combinatorial Optimization with RL

백승언

08 Nov, 2021

Contents

Introduction

Combinatorial optimization problems

- **What is Combinatorial Optimization(CO) problem?**
  - Finding an optimal objects from a finite set of objects[1]
  - In many hard CO problems, exhaustive search is not tractable
    - Optimal solution require exponential computing time
  - Conventional CO problems generally rely on handcrafted heuristics or optimization methods
    - Once the problem statement changes slightly, they need to be revised and this processes are time-consuming
- **Machin learning approaches to CO problem**
  - Supervised learning is not applicable to most CO problems because it is difficult to acquire optimal labels
    - Exceptionally, Pointer Network beats the CO problem with small-size problems
      - TSP-50, TSP-100, TSP-200 and so on
  - Many studies show that the Reinforcement Learning(RL) have potential to tackle the CO problems
    - This paper[2] is the beginning of such studies.

[1] : https://en.wikipedia.org/wiki/Combinatorial_optimization [2]: https://arxiv.org/pdf/1611.09940.pdf

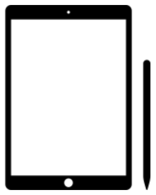
Examples of CO

- **Knapsack Problem(KP)**
  - 무게 제한 아래에서, 최고의 효용을 얻을 수 있는 물품들을 가방에 담는 문제
    - $z = \Sigma_{i=1}^{n} v_i x_i, \quad \mathrm{s.t.}\, \Sigma_{i=1}^{n} w_i x_i \le W, \quad x_i \in Z^{0+}$

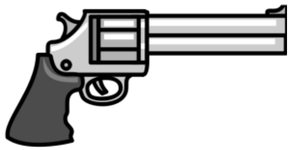
- **Travelling Salesman Problem(TSP)**
  - 방문 판매원이 최소 비용으로 모든 지역을 순회하는 방법에 관해 묻는 문제
    - $z = \left\|x_n - x_1\right\|_2 + \Sigma_{i=1}^{n-1} \left\|x_i - x_{i+1}\right\|_2$

Notebook
4kg, 8 happiness

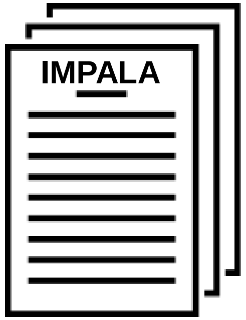
Beverage
1kg, 4 happiness



Tablet PC
3kg, 6 happiness



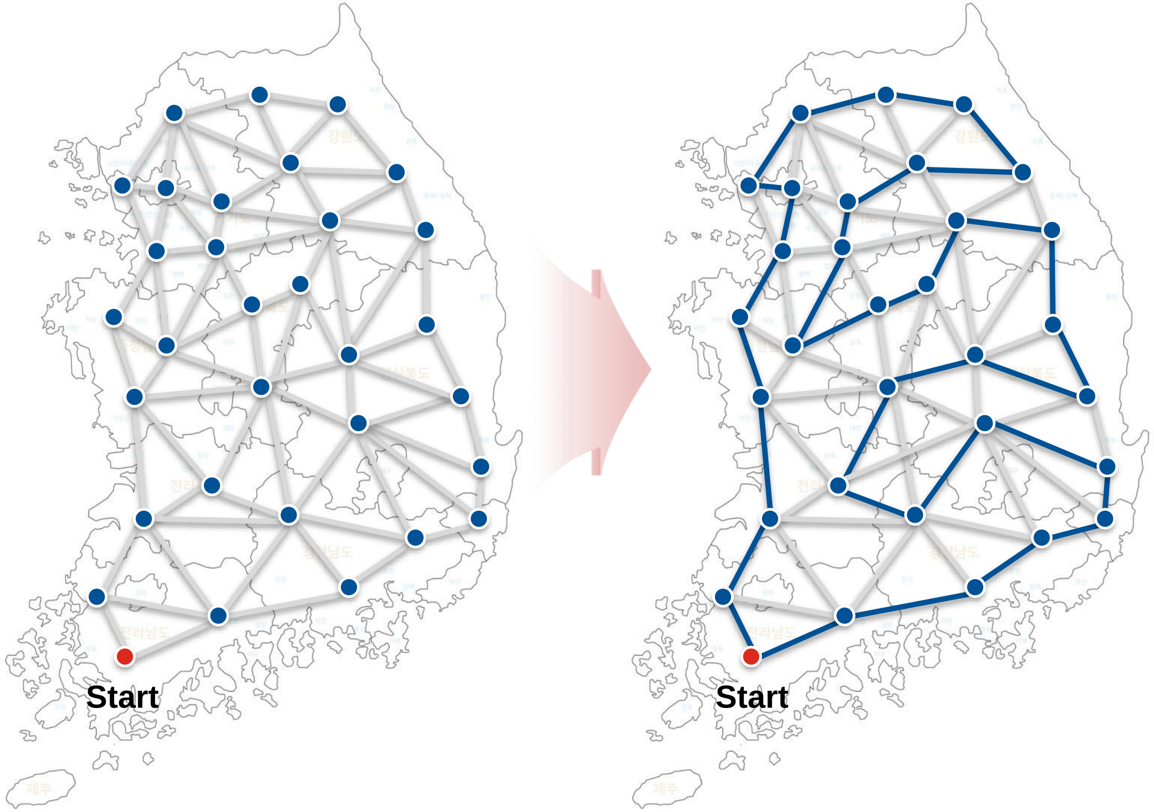
Revolver
3kg, 3 happiness



RL papers
1kg, 1 happiness

Camera
2kg, 4 happiness

Maximum weight: 8 Kg



Neural Combinatorial Optimization with RL

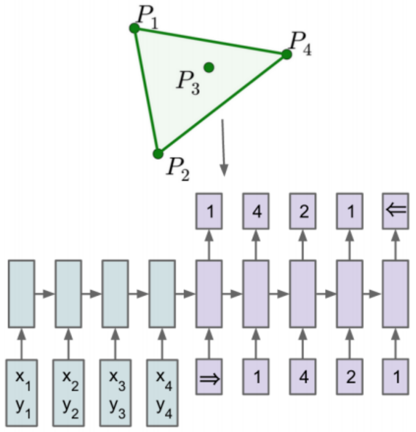
Backgrounds

- **The difficulty in applying existing search heuristics to newly encountered problem**
  - All search algorithms require the **prior knowledge** over problems to guarantee performance
  - Requirements for generalized solution at which can handle the various CO problems has increased
    - Oh, can neural network do this?

- **Previous application of neural networks to combinatorial optimization**
  - Hopfield network
    - Hopfield proposed Hopfield networks to solve the TSP (1985)
    - Limitation of Hopfield network was issued and resolved by Wilson & Pawley (1988)
  - Deformable template model
    - Durbin proposed the elastic network to solve TSP (1987)
    - The application of Self Organizing Map to TSP was proposed by Fort, Angeniol, Kohonen, (1988~1990)
  - Seq2Seq learning
    - Yuitan et al, Zoph & Le purposed the study for optimization in various domain. (2016)
    - Vinyals et al revisited the TSP by their **Pointer Network** (2015)

Important previous study: Pointer network

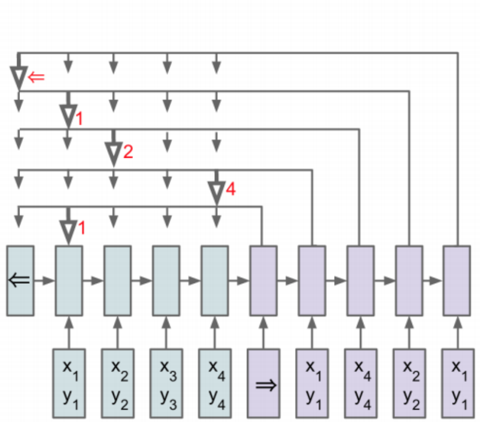
- **A network that can adjust the output length**
  - Previous recurrent network cannot handle the variable frame
    - Seq2Seq paradigm beaten this issue using **encoder-decoder structure**
  - Seq2Seq, however, has limitation that it cannot adjust the variable output length
    - The Pointer Network resolve this issue by simple idea
  - The Pointer Network utilize the attention score of the input embedding vector
    - It seems like decoder output **point** the input vector



$$
\begin{aligned}
u_j^i &= v^T \tanh(W_1 e_j + W_2 d_i) \\
a_j^i &= \mathrm{softmax}(u_j^i) \\
d_i' &= \sum_{j=1}^{n} a_j^i e_j
\end{aligned}
$$

$$p(\mathcal{C}^{\mathcal{P}}|\mathcal{P};\theta) = \prod_{i=1}^{m(\mathcal{P})} p_\theta(C_i|C_1,\ldots,C_{i-1},\mathcal{P};\theta).$$

Seqeunce-to-Sequence / Content-based input attention

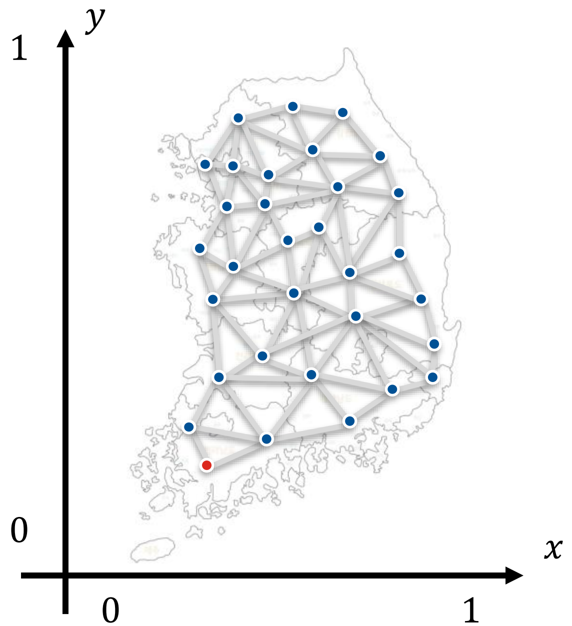


$$
\begin{aligned}
u_j^i &= v^T \tanh(W_1 e_j + W_2 d_i) \\
p(C_i|C_1,\ldots,C_{i-1},\mathcal{P}) &= \mathrm{softmax}(u^i)
\end{aligned}
$$

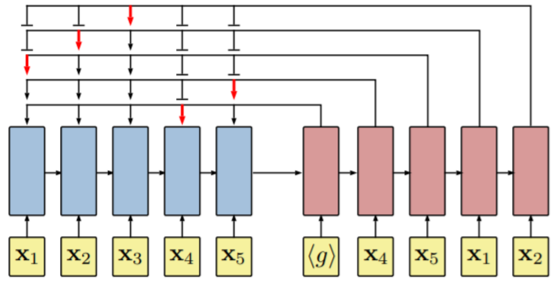
Pointer Network

https://arxiv.org/abs/1506.03134

- **Neural network architecture for TSP**
  - Problem definition
    - State
      - sequence of n cities in a 2D Euclidean space
      - $s = \{\mathbf{x}_i\}_{i=1}^{n}, \quad \mathbf{x}_i \in \mathbb{R}^2$
    - Objective function of TSP(return G)
      - length of a tour defined by a permutation $\boldsymbol{\pi}$
      - $L(\pi \mid s) = \left\|\mathbf{x}_{\pi(n)} - \mathbf{x}_{\pi(1)}\right\|_2 + \Sigma_{i=1}^{n-1}\left\|\mathbf{x}_{\pi(i)} - \mathbf{x}_{\pi(i+1)}\right\|_2, \quad \|\cdot\|\ denots\ l_2\ norm$
    - Policy
      - Using Pointer network
      - Chain rule to factorize the probability of a tour as
      - $p(\pi \mid s) = \prod_{i=1}^{n} p(\pi(i) \mid \pi(< i), s)$
    - Pointing mechanism
      - $u_i = \begin{cases} v^T \cdot \tanh(W_{ref} \cdot r_i + W_q \cdot q) & if\ i \neq \pi(j) for\ all\ j < i \\ -\infty & \end{cases} \quad for\ i = 1, 2, \dots, k$
      - $A(ref, q; W_{ref}, W_q, v) \triangleq softmax(C \tanh(\boldsymbol{u})), \quad C\ is\ a\ hyperparameter$



TSP on Euclidean space



Pointer network architecture

- **Neural network architecture for TSP**
  - Policy optimization
    - Actor
      - Using Pointer network
      - Mapping input sequence $\boldsymbol{s}$ and previous action set $\{a_j\}, j \in \{1, \dots, i-1\}$ to action a_i
    - Critic
      - Using Pointer network
      - Mapping input sequence $\boldsymbol{s}$ to baseline $b_{\theta_v}(\boldsymbol{s})$
    - Loss function of actor, critic
      - $J(\boldsymbol{\theta} \mid s) = \mathbb{E}_{\pi \sim p_\theta(\cdot|s)} L(\pi \mid s), \quad s \in \mathcal{S}$
      - $\nabla_\theta J(\theta \mid s) = \mathbb{E}_{\pi \sim p_\theta(\cdot|s)}[(L(\pi \mid s) - b(s)) \nabla_\theta \log p_\theta(\pi \mid s)]$, (**REINFORCE**)
      - $L(\theta_v) = \frac{1}{B} \Sigma_{i=1}^{B} \left\| b_{\theta_v}(s_i) - L(\pi_i \mid s) \right\|_2^2$

Algorithm 1 Actor-critic training

procedure TRAIN(training set S, number of training steps T, batch size B)
  Initialize pointer network params θ
  Initialize critic network params θ_v
  for $t = 1$ to T **do**
    $s_i \sim \text{SAMPLEINPUT}(S)$ for $i \in \{1, \dots, B\}$
    $\pi_i \sim \text{SAMPLESOLUTION}(p_\theta(.|s_i))$ for $i \in \{1, \dots, B\}$
    $b_i \leftarrow b_{\theta_v}(s_i)$ for $i \in \{1, \dots, B\}$
    $g_\theta \leftarrow \frac{1}{B} \sum_{i=1}^{B} (L(\pi_i|s_i) - b_i) \nabla_\theta \log p_\theta(\pi_i|s_i)$
    $\mathcal{L}_v \leftarrow \frac{1}{B} \sum_{i=1}^{B} \|b_i - L(\pi_i)\|_2^2$
    $\theta \leftarrow \text{ADAM}(\theta, g_\theta)$
    $\theta_v \leftarrow \text{ADAM}(\theta_v, \nabla_{\theta_v} \mathcal{L}_v)$
  end for
  return θ
end procedure

Pseudo code of actor-critic training

Methodology (III)

- **Search strategies**
  - RL pretraining - Greedy
    - After training, select the tours from $p_\theta(\cdot|\, s)$ and greedy method
  - RL pretraining - Sampling
    - After training, select the multiple candidate tours(1.28M) from stochastic policy $p_\theta(\cdot|\, s)$ and select the shortest one
    - Controlling the diversity of the sampled tour with a temperature hyperparameter T
      - $A(ref, q, T; W_{ref}, W_q, v) \triangleq softmax(u/T)$
  - Active Search
    - Refining the parameters of stochastic policy $p_\theta(\cdot|\, s)$ during inference to minimize $J(\theta)$ on a single test input **s**
      - Only policy updated, not critic.
  - RL pretraining - Active Search
    - After training, using active search

Algorithm 2 Active Search

procedure ACTIVESEARCH(input s, θ, number of candidates K, B, α)
  $\pi \leftarrow$ RANDOMSOLUTION()
  $L_\pi \leftarrow L(\pi \mid s)$
  $n \leftarrow \lceil \frac{K}{B} \rceil$
  for $t = 1 \ldots n$ **do**
    $\pi_i \sim$ SAMPLESOLUTION($p_\theta(. \mid s)$) for $i \in \{1, \ldots, B\}$
    $j \leftarrow$ ARGMIN($L(\pi_1 \mid s) \ldots L(\pi_B \mid s)$)
    $L_j \leftarrow L(\pi_j \mid s)$
    if $L_j < L_\pi$ **then**
      $\pi \leftarrow \pi_j$
      $L_\pi \leftarrow L_j$
    end if
    $g_\theta \leftarrow \frac{1}{B} \sum_{i=1}^{B} (L(\pi_i \mid s) - b) \nabla_\theta \log p_\theta(\pi_i \mid s)$
    $\theta \leftarrow$ ADAM(θ, g_θ)
    $b \leftarrow \alpha \times b + (1 - \alpha) \times (\frac{1}{B} \sum_{i=1}^{B} b_i)$
  end for
  return π
end procedure

Pseudo code of active search

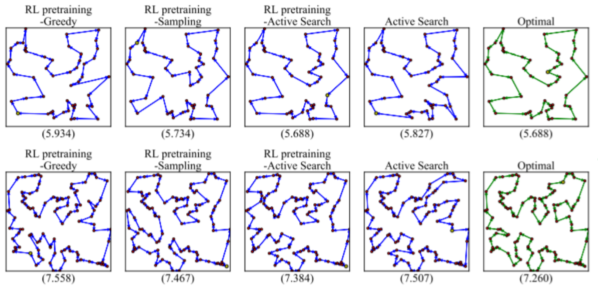
Configuration	Learn on training data	Sampling on test set	Refining on test set
RL pretraining-Greedy	Yes	No	No
Active Search (AS)	No	Yes	Yes
RL pretraining-Sampling	Yes	Yes	No
RL pretraining-Active Search	Yes	Yes	Yes

Different learning configurations

Experiment Results

Experiment results (I)

- **Comparison with different strategies for TSP family**
  - 3 개의 TSP에 대해서, 논문에서 제시한 기법과, 기존에 사용되던 기법들, 최적 해 사이의 성능을 비교하는 실험을 수행(TSP20, TSP50, TSP100)
    - Supervised learning(Pointer network)
    - RL pretraining - greedy
    - RL pretraining - sampling
    - RL pretraining - active search
    - Active search
    - Christofides: polynomial time complexity
    - OR Tools' local search: made by Google
    - Optimal solution
  - 각 기법들을 비교할 성능 지표로는, Average tour length와 running time을 선택
    - RL+sampling, RL+AS의 경우, optimal에 근접한 결과를 얻어냈음을 확인
    - RL+greedy의 경우, 다른 기법들과 비교해 보았을 때 , 시간 효율적임을 확인



***Sample results.* TSP50, TSP100**

Task	Supervised Learning	RL pretraining				AS	Christo -fides	OR Tools' local search	Optimal
		greedy	greedy@16	sampling	AS				
TSP20	3.88(†)	3.89	–	3.82	3.82	3.96	4.30	3.85	3.82
TSP50	6.09(†)	5.95	5.80	5.70	5.70	5.87	6.62	5.80	5.68
TSP100	10.81	8.30	7.97	7.88	7.83	8.19	9.18	7.99	7.77

Average tour length of different methods

Task	RL pretraining		OR-Tools' local search	Optimal	
	greedy	greedy@16		Concorde	LK-H
TSP50	0.003s	0.04s	0.02s	0.05s	0.14s
TSP100	0.01s	0.15s	0.10s	0.22s	0.88s

Running time in seconds

Experiment results (II)

- **Comparison with different strategies for KnapSack problems**
  - 3 개의 KanpSack problem에 대해서, 논문에서 제시한 기법 일부와, 기존에 사용되던 기법들, 최적 해 사이의 성능을 비교하는 실험을 수행(KNAP50, KNAP100, KNAP200)
    - RL pretraining - greedy
    - Active Search
    - Random search
    - Greedy
    - Optimal solution
  - 각 기법들을 비교할 성능 지표로는, total value z로 설정
    - RL+greedy의 경우, optimal solution의 1% 이하의 오차율을 보였음을 확인
    - Active Search의 경우, 최적해를 찾았음을 확인

Task	RL pretraining greedy	Active Search	Random Search	Greedy	Optimal
KNAP50	19.86	**20.07**	17.91	19.24	**20.07**
KNAP100	40.27	**40.50**	33.23	38.53	**40.50**
KNAP200	57.10	**57.45**	35.95	55.42	**57.45**

Total value of different methods

Thank you!

Q&A

Implementation of NCO

Environment setting

- **Environment: OR-gym**
  - Knapsack-v0
    - 무게 제한 아래에서, 최고의 효용을 얻을 수 있는 물품들을 가방에 담는 문제
    - $z = \Sigma_{i=1}^{n} v_i x_i,\ \ \mathrm{s.t.} \Sigma_{i=1}^{n} w_i x_i \leq W,\ \ x_i \in Z^{0+}$
  - MDP formulation
    - State:
      - [Item weights, item values, current weight]
    - Action:
      - Number of item
    - Reward:
      - Item value : value of item
      - Over packed penalty : -100
  - TSP-v0
    - 최소 비용으로 모든 지역을 순회하는 방법에 관해 묻는 문제
    - $z = ||x_n - x_1||_2 + \Sigma_{i=1}^{n-1} ||x_i - x_{i+1}||_2$
  - MDP formulation
    - State:
      - [current node, connection(adjacency matrix)]
    - Action:
      - Number of node
    - Reward:
      - Cost of move : -distance
      - Invalid action penalty : -100
      - Success : 1000

Result analysis

- **LunarLanderContinuous-v2**
  - Comparison with Vanilla TD3(**TD3_None**), TD3_gSDE(8, 16, 32, **64, Episode**)
    - Figure 1 shows that TD3_gSDE is better stability than Vanilla TD3

실험 중

Learning environment