A Neural Network Solver for Local Search in Genetic Algorithm to solve Travelling Salesman Problem

Jinchang Fan, Han Wang, Xiaoyu Qiao, Ziyan Lin

March 28, 2019

Abstract

Many solutions for Traveling Salesman Problem have been raised and discussed for a long time, and one of the best approximate solutions is Genetic Algorithm for global search combined with Lin-Kernighan Heuristic for local search. Though accuracy is satisfying, the efficiency of this method is limited by the iteration of Lin-Kernighan Heuristic to reach a local optimal. In this project, the cutting-edge Deep Neural Network is applied to reach the local optimal in one step which would increase the efficiency to find a solution.

1 Introduction

Traveling Salesman Problem is a famous optimization problem that has been discussed for a long time. The problem asks: given a set of cities and cost of travelling from one to another, what's best way in terms of cost to visit all cities and come back to the starting point? In contract to its simple description, TSP is found to be an NP-hard problem[1]. Considering its popularity and various real-world applications, TSP now performs as a touchstone for different optimizing algorithms.

So far, algorithms for exact solution of TSP[2][3][4] have been widely developed and discussed respectively, but the solution is still infeasible to reach when the number of cities grows large due to the exponential complexity[5].

At the same time approximate algorithms[6][7] are developed to obtain a acceptable solution in relatively short time, including many bioinspired algorithms[8][9][10]. Genetic Algorithm[11] is one of the best algorithms in many combinatorial optimization problems with huge research work thanks to its efficient strategy to reach global optimization, and has been shown to perform quite good on TSP[12]. The basic procedure of GA on TSP could be described as following: first start with a random initial population and apply local optimizing strategy for each of them, then select instance as "parents" to generate new off-springs by cross-over. Apply local optimizing strategy again to those off-springs, and partially update population by off-springs after mutation. Repeat generations and population would converge to the global optimization. In the Genetic Algorithm for TSP, local search strategy is critical.

A current popular local search method is Lin-

Kernighan algorithm[13][5] with basic idea of performing exchanges of links to obtain a shorter tour. The algorithm is shown in detail in Section 2.2. It is a quite simple idea but usually the simplicity of the local search algorithm is obtained with the cost of the possibility of failure in finding optimal solutions. For example, the algorithmic structure of the modified Lin-Kernighan algorithm of Mak and Morton [14] is very simple. However, the algorithm cannot even guarantee 2-opt optimality. It is also of interest to reduce the running time. Keld Helsgaun[5] showed that the running time is approximately $O(n^{2.2})$. It would be time consuming when solving large problems.

Deep Neural Network (DNN)[15] is a prediction method with multiple layers connected to nonlinear operators is the rectified linear unit (ReLU), whose output is the maximum of its input value and zero. Such structure has great flexibility to model complex non-linear relationships. Finding the local optimal tour is a quite complicated problem for Lin-Kernighan algorithm, but it may be easy for DNN model to simulate the process. Once an ideal DNN model is found to predict the local optimal, The time consuming work for Lin-Kernighan algorithm could be saved, and efficiency of the whole Genetic Algorithm could be improved consequently. Such improvement would be the main focus of this project. Methodology would be discussed and practice on sample case are performed. The new algorithm is compared with previous work in the evaluation of accuracy and time consumed.

The rest of this report is organized as follows. Section 2 makes a detailed description of the GA-DNN algorithm. Section 3 introduces data sets which algoritms is applied to, and Section 4 evaluates the result of experiments, including accuracy and time efficiency. The whole project is summarized in Section 5.

2 GA-DNN algorithm for TSP

In this project the structure of Genetic Algorithm is combined with Deep Neural Network for local search algorithm. Traditional Lin-Kernighan algorithm for local search is applied to train a Neural Network. Details in the procedure is shown in following subsections.

2.1 Genetic Algorithm

Genetic Algorithm (GA) is an important tool in solving both constrained and unconstrained optimization problem with inspiration of natural selection. The procedure of Genetic Algorithm in the context of TSP is illustrated in Fig 2.1. Some important definitions are stated as below.

- Tour: a route that visits each city exactly one time and returns to the starting point in the end
- Population: a collection of possible tours
- Parents: two tours that could be used to breed children tour
- Mating pool: a collection of parents that are used to create our next population of children tours
- Fitness: measure for goodness of tours
- Mutation: variation in tours' population
- Elitism: keep best k tours into the next generation

procedure TSP-GA
begin
  `initialize population P randomly`
  foreach `tour` $t \in P$ do $LocalSearch(t)$;
  repeat
    for $i = 0$ to #`new tours` do
      `randomly select` $t_{p1}, t_{p2} \in P$;
      $t_c := breed(t_{p1}, t_{p2})$;
      $LocalSearch(t_c)$;
      $Mutation(t_c)$;
      `replace one tour in P by` t_c;
    end;
  until `converged`;
end

Figure 2.1. Genetic Algorithm for TSP

In the breeding part, there are many crossover methods could be used, including "ordered crossover" and "distance preserved crossover". In this work "distance preserved crossover" is used to do the breeding.

In the mutation part, two cities (points) in tour are randomly swapped to generate variation in the population.

2.2 Lin-Kernighan Heuristic

The idea of the basic Lin-Kernighan algorithm is performing exchanges of links to obtain a shorter tour. The algorithm is shown in detail in Fig 2.2.

Lin-Kernighan Algorithm
1. Generate a random initial tour T.
2. Let $i = 1$. Choose t_1.
3. Choose $x_1 = (t_1, t_2) \in T$.
4. Choose $y_1 = (t_2, t_3) \in T$ such that $G_1 > 0$.
   If this is not possible, go to Step 12.
5. Let i = i+1.
6. Choose $x_i = (t_{2i-1}, t_{2i}) \in T$ such that
   (a) if t_{2i} is joined to t_1 , the resulting
   configuration is a tour, T', and;
   (b) $x_i \neq y_s$ for all $s < i$.
   If T' is a better tour than T,
   let $T = T'$ and go to Step 2.
7. Choose $y_i = (t_{2i}, t_{2i+1}) \notin T$ such that
   (a) $G_i > 0$,
   (b) $y_i \neq x_s$for all $s \leq i$, and
   (c) x_{i+1} exists.
   If such y_i exists, go to Step 5.
8. If there is an untried alternative for y_2, let $i = 2$ and go to Step 7.
9. If there is an untried alternative for x_2, let $i = 2$ and go to Step 6.
10. If there is an untried alternative for y_1, let $i = 1$ and go to Step 4.
11. If there is an untried alternative for x_1, let $i = 1$ and go to Step 3.
12. If there is an untried alternative for t_1, then go to Step 2.
13. Stop (or go to Step 1).

Figure 2.2: Lin-Kernighan Algorithm

2.3 GA-DNN Procedure

In general, the GA-DNN procedure aims to use neural network in deep learning to train local search optimization in TSP. As shown in Fig 2.3, Lin-Kernighan algorithm is applied in the first k generations for local search. DNN is trained based on the local search results and then used to predict local optimal in the following generations.

procedure GA-DNN
begin with GA
  repeat k times:
  # first k generations
    for $t_{p1}, t_{p2} \in P$:
      $t_c := breed(t_{p1}, t_{p2})$;
      $t_c\prime := LKH(t_c)$;
      $Mutation(t_c\prime)$;
      replace one tour in P by $t_c\prime$;
    end;
train DNN: $DNN(t_c) \longrightarrow t_c\prime$;
repeat
 # DNN for following generations
  for $t_{p1}, t_{p2} \in P$:
    $t_c := breed(t_{p1}, t_{p2})$;
    $t_c^D := DNN(t_c)$;
    $Mutation(t_c^D)$;
    replace one tour in P by t_c^D;
  end;
until converged;
end

Figure 2.3. GA-DNN Procedure

Consideting DNN could not generate optimal solutions out of air, LKH can't be discarded in a total. There must be some examples of tours and their local optimal that serve as learning material for DNN. Those matrial, especially local optimal should be provided by LKH. After a DNN model is well trained, it could perform local search work insead of LKH.

3 Datasets and Experiments

Python and PyTorch are the main platform for this project. PyTorch is an open-source package that contains many deep learning applications and methods. It is the most popular programing language, based on Python and Torch. PyTorch has many advantages. Firstly, PyTorch is a faster speed platform, even with high-dimensional models. Secondly, PyTorch is easy

to learn, code, and debug. Thirdly, PyTorch is a dynamic approach to deal with graphs. Last, PyTorch contributes to increase productivity of development and decrease number of errors. Therefore, most models and applications will be applied in Python and PyTorch.

The dataset comes from homepage of Discrete and Combinatorial Optimization from Heidelberg University [1]. It totally has more than 50 graphs, and for each graph, there are two zip packages. One package contains all original coordinate information of every city, and one package contains the optimal solution that generated by other algorithms. For example, in package 'gr666.tsp.gz', it includes 666 cities' coordinate information. The first column is index of cities; the second column is x-coordinate; and the third column is y-coordinate. In package 'gr666.opt.tour.gz', it has the optimal order of these cities. Hence, in this project, each original dataset is used to check models, and all outputs obtained by models will be compared with the given optimal solution.

4 Evalution

To be added.

5 Summary

To be added.

References

[1] Michael R Gary and David S Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. 1979.

[2] Michel L Balinski and Richard E Quandt. "On an integer program for a delivery problem". In: *Operations Research* 12.2 (1964), pp. 300–304.

[3] Samuel Eilon et al. "Distribution management-mathematical modelling and practical analysis". In: *IEEE Transactions on Systems, Man, and Cybernetics* 6 (1974), pp. 589–589.

[4] Gilbert Laporte, Hélène Mercure, and Yves Nobert. "An exact algorithm for the asymmetrical capacitated vehicle routing problem". In: *Networks* 16.1 (1986), pp. 33–46.

[5] Keld Helsgaun. "An effective implementation of the Lin–Kernighan traveling salesman heuristic". In: *European Journal of Operational Research* 126.1 (2000), pp. 106–130.

[6] Geoff Clarke and John W Wright. "Scheduling of vehicles from a central depot to a number of delivery points". In: *Operations research* 12.4 (1964), pp. 568–581.

[7] Billy E Gillett and Leland R Miller. "A heuristic algorithm for the vehicle-dispatch problem". In: *Operations research* 22.2 (1974), pp. 340–349.

[8] Natalio Krasnogor, Jim Smith, et al. "A Memetic Algorithm With Self-Adaptive Local Search: TSP as a case study." In: *GECCO*. 2000, pp. 987–994.

[9] Licheng Jiao and Lei Wang. "A novel genetic algorithm based on immunity". In: *IEEE Transactions on Systems, Man, and Cybernetics-part A: systems and humans* 30.5 (2000), pp. 552–561.

[10] Noraini Mohd Razali, John Geraghty, et al. "Genetic algorithm performance with different selection strategies in solving TSP". In: *Proceedings of the world congress on engineering*. Vol. 2. 1. International Association of Engineers Hong Kong. 2011, pp. 1–6.

[11] Darrell Whitley. "A genetic algorithm tutorial". In: *Statistics and computing* 4.2 (1994), pp. 65–85.

[12] Peter Merz and Bernd Freisleben. "Genetic local search for the TSP: New results". In: *Proceedings of 1997 Ieee International Conference on Evolutionary Computation (Icec'97)*. IEEE. 1997, pp. 159–164.

[1] https://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/

[13] Shen Lin and Brian W Kernighan. “An effective heuristic algorithm for the traveling-salesman problem”. In: *Operations research* 21.2 (1973), pp. 498–516.

[14] King-Tim Mak and Andrew J Morton. “A modified Lin-Kernighan traveling-salesman heuristic”. In: *Operations Research Letters* 13.3 (1993), pp. 127–132.

[15] Simon Haykin. *Neural networks*. Vol. 2. Prentice hall New York, 1994.