An iterated local search for the travelling salesman problem

Pedro B. Castellucci[1]

[1]Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brazil

{pedro.castellucci}@ufsc.br

Abstract. *The Traveling Salesman Problem is a classical problem in Computer Science, with many extensions and variations being studied along decades of research, particularly for applications related to logistics. Here, we present a Iterated Local Search (ILS) algorithm to find solutions for instances of the problem as proposed by The Brazilian competition on Knowledge Discovery in Databases (KDD-BR). The ILS algorithm is a simple algorithm that does not depend on any third-party libraries. Also, it was among most effective methods in the competition.*

1. Introduction

The Traveling Salesman Problem (TSP) is one of the most studied problems in Computer Science. The problem can be defined based on a graph $G = (V, E)$ in which V is the set of nodes (e.g. cities) and E the set of edges connecting these nodes. Associated with each edge there is a cost $c_{ij} \geq 0$. The goal is to find a tour $r = (i_1, i_2, \ldots, i_{|V|}, i_1)$ which visits each node exactly once and has a minimum cost.

Despite the NP-hardness of the problem, state-of-the-art exact methods can currently solve, with proved optimality, instances with several thousands of nodes. These methods are based on the branch-and-cut framework from integer programming [Applegate et al. 2011]. Regarding heuristic methods, LKH-based heuristics achieve state-of-the-art results [Tinós et al. 2018]. One important component of LKH heuristic is the k-Opt movements, which we used for this competition. These moves are well known in the TSP literature [Applegate et al. 2011] and for other routing problems.

More recently, there has been an emergent trend of applying machine learning techniques to combinatorial NP-hard problems, particularly for the TSP ([Ishaya et al. 2019], [Joshi et al. 2019] and [Vitali et al. 2021]). There have been some interesting results but these techniques are yet to be competitive with state-of-the-art methods.

The goal of the competition was presented as follows: *"The objective of this challenge is to develop a method to predict the edges belonging to solutions of traveling salesman problems."*

Aiming at this goal, we implemented a solution based on a metaheuristic known as *Iterated Local Search* (ILS). Then, we use the solution provided by the ILS as a predictor of the edges in an optimal solution. As an abstract method, this is similar to techniques such as Neural Networks. It tries to find a solution that optimizes a function (the cost of a route, in our ILS idea) and uses this solution to define a prediction.

The ILS algorithm is described in Section 2, with its components in subsections 2.1 and 2.2. Then, we offer some comments regarding the computational experiments and implementation in Section 3.

2. Iterated Local Search

The Iterated Local Search is a simple metaheuristic but with competitive results for complex combinatorial problems [Lourenço et al. 2019]. It is also flexible, being applied not only to routing problems (e.g. [Stützle and Hoos 2002], [Vansteenwegen et al. 2009] and [Penna et al. 2013]) but to others such as timetabling problems (e.g. [Song et al. 2018]) and packing problems (e.g. [Imamichi et al. 2009]).

To apply the ILS metaheuristic, a problem specific algorithm should be known, then it is used as a local search procedure. This local search is combined with a perturbation phase to restart the search with the intent of avoiding local optimal solutions. Algorithm 1 presents a pseudo-code of our ILS based solution.

Algorithm 1: Iterated local search

Data: An initial solution s_0 and a maximum number of iterations $maxIter$.
Result: A solution s^*.

```
1 s*: LocalSearch(s_0) /* See Subsection 2.1.                    */
2 iter = 0
3 while iter < maxIter do
4 |   s' = Perturbation(s*, iter) /* See Subsection 2.2          */
5 |   s*' = LocalSearch(s')
6 |   if f(s*') < f(s*) then
7 |   |   s* = s*'
8 |   |   iter = 0
9 |   iter = iter + 1
```

The ILS algorithm has as an input an initial solution. For instances with n nodes, we used $(1, 2, \ldots, n)$ which is sorted as provided by the challenge organization. The first step (line 1 in Algorithm 1) is a local search based on k-opt movements, described in Subsection 2.1. After the local search finds a local optimum, the main loop (lines 3–9) alternates between a perturbation phase (line 4) and a local search (line 5). The perturbation procedure tries to escape the local optimal by disturbing the current solution (details in Subsection 2.2). Then, lines 6–8, account for saving the best solution found so far. Every time there is a solution improvement, the iteration count is reset (line 8).

2.1. Local search phase

For the local search phase, we used k-opt movements, with $k = 2$ and $k = 3$. A k-opt movement consists of replacing k edges of a solution for different ones. For the 2-opt move, for example, there is only one way to recreate a tour (Figure 1). For the 3-opt movements there are seven different ways (three of them equivalent to 2-opt moves), we try each combination to find the one with the least cost.

The local search phase is a maximum descent algorithm. It searches for the pair of edges the minimize the total cost of the tour when applying a 2-opt movement. Then,

(a) Before a 2-opt move. **(b) After a 2-opt move.**

Figure 1. Example of a 2-opt move, removing edges $(i, i+1)$ and $(j, j+1)$.

it searches for the three edges that minimize the total cost of the tour when applying a 3-opt movement. After the maximum descent algorithm, the solution is a local optimum in the neighborhood defined by the 3-opt movements. Finally, we perturb the solution (Subsection 2.2) to try to escape this local optimum.

Note that, even though 2-opt movements are included in 3-opt movements, we first apply 2-opt followed by the remaining 3-opt moves. From empirical observations, this resulted in a faster algorithm, probably because of the reduced number of 3-opt movements needed to reach a local optimal.

2.2. Perturbation phase

To define the perturbation strategies, consider a tour r. Let $r[i]$ be the i-th city in the tour and $r[i:j]$ being the path from i to j (both included). We defined three strategies: *(i)* Randomly sample i from $[1 : n - gap]$ and j from $[i + gap, n]$ and shuffle the cities in path $r[i : j]$. *(ii)* Randomly sample i from $[1 : n - gap]$ and j from $[i + gap, n]$, shuffle the cities in path $r[i : j]$ and randomly reposition path $r[i : j]$ along the route. The gap parameter is used as 30 in our solution. Finally, *(iii)* shuffle route r completely.

Each perturbation strategy is applied at different moments during the search. Strategy *(i)* is applied when $0 \leq iter < 400$, strategy *(ii)* when $400 \leq iter < 1500$ and strategy *(iii)* when $iter \geq 1500$ (see Algorithm 1).

The algorithm terminates when the maximum number of iterations without any improvement is reached ($maxIter = 2500$). The edges in the best route found s^* are then used as a predictor for the edges in an optimal solution, i.e., if (i, j) is part of route s^* then we assume is part of the optimal solution.

3. Computational remarks

According to challenge rules, the metric for evaluating the results is the F1 metric of the predicted adjacency matrix compared to the optimal solution of each of the 2000 instances. This metric was computed on a fraction of the test set and the result was 0.07906 for the Iterated Local Search based strategy presented in Section 2.

The source code is available at `https://gitlab.com/pbcastellucci/bracis-kdd-tsp-challenge-2021`. It was implemented in about 300 lines of Julia code and does not depend on any third-party library. Regarding the computational enviroment, we used Linux Mint 20 OS and ran the experiments in an Intel(R) Core^{TM} i5-8265U 4x1.60GHz, 8GB of RAM.

Regarding efficiency, the prediction is generated in seconds for each instance. The size of the instances varies from 50 to 200 city nodes. Due to the nature of the competition, we did not focus on precisely measuring the computational time nor the contribution of each component of the solution, which could be interesting for better characterizing the algorithm. Also, there are potential improvements to be made in tuning the parameters of the algorithm. This tuning could use the training set provided.

Acknowledgments

We thank the organizers of the 5th edition of the Brazilian competition on Knowledge Discovery in Databases, the reviewers and the sponsors of the event.

References

Applegate, D. L., Bixby, R. E., Chvátal, V., and Cook, W. J. (2011). *The traveling salesman problem*. Princeton university press.

Imamichi, T., Yagiura, M., and Nagamochi, H. (2009). An iterated local search algorithm based on nonlinear programming for the irregular strip packing problem. *Discrete Optimization*, 6:345–361.

Ishaya, J., Ibrahim, A., and Lo, N. (2019). A comparative analysis of the travelling salesman problem: Exact and machine learning techniques. *Open Journal of Discrete Applied Mathematics*, 2:23–37.

Joshi, C. K., Laurent, T., and Bresson, X. (2019). An efficient graph convolutional network technique for the travelling salesman problem. *arXiv preprint arXiv:1906.01227*.

Lourenço, H. R., Martin, O. C., and Stützle, T. (2019). Iterated local search: Framework and applications. In *Handbook of metaheuristics*, pages 129–168. Springer.

Penna, P. H. V., Subramanian, A., and Ochi, L. S. (2013). An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics*, 19:201–232.

Song, T., Liu, S., Tang, X., Peng, X., and Chen, M. (2018). An iterated local search algorithm for the university course timetabling problem. *Applied Soft Computing*, 68:597–608.

Stützle, T. and Hoos, H. H. (2002). Analysing the run-time behaviour of iterated local search for the travelling salesman problem. In *Essays and Surveys in Metaheuristics*, pages 589–611. Springer.

Tinós, R., Helsgaun, K., and Whitley, D. (2018). Efficient recombination in the lin-kernighan-helsgaun traveling salesman heuristic. In *International Conference on Parallel Problem Solving from Nature*, pages 95–107. Springer.

Vansteenwegen, P., Souffriau, W., Berghe, G. V., and Van Oudheusden, D. (2009). Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research*, 36:3281–3290.

Vitali, T., Mele, U. J., Gambardella, L. M., and Montemanni, R. (2021). Machine learning constructives and local searches for the travelling salesman problem. *arXiv preprint arXiv:2108.00938*.