Seminar 2

Traveling Salesman Problem Solvers

Mathew Thiel

2-16-2024

Contents

1 Introduction

The first mention of the Traveling Salesman Problem (TSP) in mathematic literature came from Karl Menger after a 1930 mathematics colloquium [1]. The problem was formal formalized by A.W. Tucker and Merrill Flood at Princeton in context of Flood's school-bus routing study [2]. Flood moved on to work at the RAND Corporation, which produced a breakthrough in TSP litature as Dantzig, Fulkerson, and Johnson published a method for solving the TSP with 49 cities. Seventeen years later, Held and Karp were next to produce a standard in TSP calculation with their branch and bound method [1]. In 1990, Gerhard Reinelt created TSPLIB, which is a, library of TSP challenge instances up to 85,900 cities [1]. Applegate, Bixby, Chvatal, and Cook used the cutting plane method to create the Concorde solver, which was able to solve all 34 instances of TSPLIB [1]. Concorde, as of 2010, is considered "the best available solver for the symmetric TSP" [3]. Modern implementations of the TSP use methods ranging from 2-OPT, Genetic Algorithms, Particle Swarm, and Tabu Search. This seminar covers Cutting Plane, Branch and Bound, Genetic Algorithms, and Particle Swarm for solving TSP.

2 Solvers

2.1 Branch and Bound

Is a mathematical solver that uses a "divide and conquer" method to solve problems by putting bounds on the optimal cost to avoid exploring parts of the problem space [4]. Branch and bound methods essentially form sub-problems where a lower bound is obtained instead of finding the optimal cost exactly. For some sub-problems, an optimal is found so that an upper bound can be created [4]. If the lower bound for a sub-problem does not improve the best found so far, then that sub-problem is discarded. The steps for a general branch and bound are as follows[4]:

1. Select an active sub-problem F_i
2. If the sub-problem is infeasible delete; otherwise, compute $b(F_i)$ for the corresponding sub-problem.
3. If $b(F_i) \geq U$, delete the sub-problem.
4. If $b(F_i) < U$, either obtain an optimal solution to the sub-problem or break the corresponding sub-problem into further sub-problems, which are added to the list of active sub-problems.

2.2 Cutting Plane

The cutting plane algorithm essentially takes an integer programming problem and "cuts" it into smaller problems with inequalities. The problem is first relaxed, then a solution is found. If the solution is an integer, then it is the optimal solution. If not, an inequality is found that the solution does not satisfy but the solutions for the original integer problem

do. Then, that inequality is added to the linear programming problem to form a tighter relaxation, and this adding of inequalities is iterated over [4]. The integer programming problem is defined as[4]:

$$\begin{aligned} \min \quad & c'x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \\ & x \text{ integer} \end{aligned} \tag{1}$$

And the relaxation is [4]:

$$\begin{aligned} \min \quad & c'x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{2}$$

The formal steps for the algorithm are as follows[4]:

1. Solve the linear programming relaxation in (2). Let x^* be an optimal solution.
2. If x^* is an integer stop; x^* is an optimal solution to (1)
3. If not, add a linear inequality constraint to (2) that all integer solutions to (1) satisfy, but x^* does not; go to Step 1.

2.3 Genetic Algorithms

Genetic algorithms are the most widely known type of evolutionary algorithms and they were initially created by J.H. Holland in his research *Adaptation in Natural and Artificial Systems* [5]. A seminal book on the subject is *Genetic Algorithms in Search, Optimization, and Machine Learning*[6], which helped to define the classical genetic algorithm, or simple

GA [5]. GAs essentially work as follows: given an initial population, parents are selected randomly after the population is shuffled. A crossover operation occurs, forming children pairs who then individually mutate and form the next generation, replacing the previous [5]. There are many variations to the simple genetic algorithm and crossover strategies, such as the Partially Modified Crossover (PMX), order crossover, matrix crossover, and heuristic crossovers. Bryant in *Genetic Algorithms and the Travelling Salesman Problem*[7] finds that matrix representation with a matrix or heuristic crossover may be best suited for TSP.

2.4 Particle Swarm Optimization

Particle Swarm Optimization (PSO) was first introduced by Eberhart and Kennedy[8] and borrows ideas from bird flocks and other communal occurrences in nature [9]. Unlike genetic algorithms, PSO doesn't rely on crossovers or mutations but instead uses shared information between "particles" and trajectories to explore the problem space. PSO uses a large number of parameters and is thus highly controllable. Parameters commonly coined as "c1" and "c2" are used to adjust the learning rate of the algorithm. Increasing c1 makes particles focus more on their best-found solution while increasing c2 makes particles tend more towards the current best population optima. Increasing the number of particles leads to fewer iterations to find a solution but more computation time, while too few particles run the risk of not finding a solution at all. The recommended number of particles is about 20-30, which is relatively small compared to other algorithms like Genetic Algorithms [10]. PSO is very frequently cited and is a very strong solver for the TSP and other problems alike. The problem is formulated below[11]:

$$\begin{aligned} v_i^k &= wv_i^{k+1} + c_1 r_1(\text{pbest}_i^k - x_i^k) + c_2 r_2(\text{gbest}^k - x_i^k) \\ x_i^{k+1} &= x_i^k + v_i^{k+1} \end{aligned} \tag{3}$$

In (3), r_1 and r_2 are random numbers, v_i^k is the velocity, x_i^k is the position of the particle, w is inertia, which scales the velocity down so that the particles don't move too fast, and c_1 and c_2 are the previously discussed learning parameters.

3 Conclusion

Linear Programming approaches based on cutting field and branch and bound seem to have stood the test of time in TSP solvers, but newer methods like Particle Swarm Optimization and Genetic Algorithms may prove more interesting for this project. The highly customized nature of PSO could add another interesting layer to the problem, as picking parameter values becomes a problem of its own. Genetic Algorithms seem to be a great starting point and will likely see widespread use in the project. I personally like the idea behind PSO and the separation between population and particle optima, and therefore may attempt an implementation depending on the problem constraints. However, any of the above solvers could prove interesting and sufficiently complex depending on the chosen implementation.

References

[1] David L. Applegate et al. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006. ISBN: 978-0-691-12993-8. URL: https://www.jstor.org/stable/j.ctt7s8xg (visited on 02/15/2024).

[2] Kevin M. Curtin et al. "A Comparative Analysis of Traveling Salesman Solutions from Geographic Information Systems". en. In: *Transactions in GIS* 18.2 (Apr. 2014), pp. 286–301. ISSN: 1361-1682, 1467-9671. DOI: 10.1111/tgis.12045. URL: https://onlinelibrary.wiley.com/doi/10.1111/tgis.12045 (visited on 02/16/2024).

[3] Donald Davendra. *Traveling Salesman Problem, Theory and Applications*. en. Dec. 2010. ISBN: 978-953-307-426-9. DOI: 10.5772/547. URL: https://www.intechopen.com/books/14 (visited on 02/15/2024).

[4] Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to Linear Optimization*. en. Google-Books-ID: GAFsQgAACAAJ. Athena Scientific, 1997. ISBN: 978-1-886529-19-9.

[5] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. en. Google-Books-ID: 7IOE5VIpFpwC. Springer Science & Business Media, Aug. 2007. ISBN: 978-3-540-40184-1.

[6] David Edward Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. en. Google-Books-ID: 2IIJAAAACAAJ. Addison-Wesley, 1989. ISBN: 978-0-201-15767-3.

[7] Kylie Bryant. "Genetic Algorithms and the Travelling Salesman Problem". en. In: ().

[8] J. Kennedy and R. Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95 - International Conference on Neural Networks*. Vol. 4. Nov. 1995, 1942–1948 vol.4. DOI: 10.1109/ICNN.1995.488968. URL: https://ieeexplore.ieee.org/document/488968 (visited on 02/16/2024).

[9] Xuesong Yan et al. "Solve Traveling Salesman Problem Using Particle Swarm Optimization Algorithm". en. In: 9.6 (2012).

[10] Maurice Clerc. *Particle Swarm Optimization*. en. Google-Books-ID: Slee72idZ8EC. John Wiley & Sons, Jan. 2010. ISBN: 978-0-470-39443-4.

[11] Majid Jaberipour, Esmaile Khorram, and Behrooz Karimi. "Particle swarm algorithm for solving systems of nonlinear equations". In: *Computers & Mathematics with Applications* 62.2 (2011), pp. 566–576. ISSN: 0898-1221. DOI: 10.1016/j.camwa.2011.05.031. URL: https://www.sciencedirect.com/science/article/pii/S0898122111004299 (visited on 02/16/2024).