TECNOLÓGICO DE MONTERREY®

Intelligent Systems

Ivan Guerrero

First Partial Project: Solving Traveling Salesman Problems via Artificial Intelligent Search Techniques

Yair Yolotl Pimentel Vanegas A01652823

Jorge German Reyes Garcia A01336637

Mexico, Mexico City

22 de marzo de 2020

Summary

The traveling salesman problem (TSP) has been firstly proposed as one of the mathematical problems for optimization in 1930s. TSP looks to find an optimal path for the traveling salesman to visit each of cities just one time, and then return to the starting city. Optimal path is whose total distance is minimized. (Suwannarongsri and Puangdownreong, 2012).

The heuristic methods could provide satisfactory solutions of the TSP who posses large amount of cities but the optimum solution can not be guaranteed. Artificial intelligence search techniques have been applied to solve the TPS, simulated annealing, artificial neural network, tabu search, and genetic algorithms are some of them.(Suwannarongsri and Puangdownreong, 2012).

TSP has been proposed as one of the mathematical problems in 1800s by Harmilton and Kirkman but the general formulation of TSP has been firstly lunched based on the graph theory in 1930s. (Suwannarongsri and Puangdownreong, 2012).

Let G = (V,E) be a complete undirected graph with vertices V,|V|=n, where n is the number of cities, and edges E with edge length c_{ij} for the-ij city (j,i) case in which $c_{ij} = c_{ij}$ for all cities.

TSP problem formulations

1) Equation that minimizes distance

$$\sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

2) Ensures that each city is entered from only one other city

$$\sum_{\substack{j \in V \\ j \neq i}} x_{ij} = 1, \quad i \in V$$

3) Ensures that each city in only departed to on other city

$$\sum_{\substack{i \in V \\ i \neq j}} x_{ij} = 1, \quad j \in V$$

4) Eliminates subtours

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1, \quad \forall S \subset V, S = \varnothing$$

5) Binary constraint, where $x_{ij} = 1$ if edge (i,j) in the solution, $x_{ij} = 0$ otherwise.

$$x_{ij} = 0 \text{ or } 1, \quad i, j \in V$$

The difficulty of TSP is that subtours constraints will grow exponentially as the number of city grows, making impossible to generate or store these constraints.

AI Search Algorithms

Genetic Algorithm: Has natural selection mechanism and genetic operation (Crossover and mutation techniques). It consists of 7 steps where the optimal solution is the best chromosome found in current population:

1) Randomly generate the populations.
2) Evaluate all population chromosomes via the objective function.
3) Select some chromosomes and set them to be parents.
4) Generate next generation of population by crossover and mutation.
5) Evaluate the (fitness) objective function of new populations.
6) Replace old population by new ones that more fit.
7) Once termination criteria are met, end search process; else go back to 2.

Tabu Search: Has the tabu list used to store the visited solutions and to conduct as an aspiration criteria when the local entrapment occurs. It consists of 6 steps:

1) Initialize a search space (Ω), TL = ∅, search radius (R), count, and countmax.
2) Randomly select an initial solution S0 from a certain search space Ω. Let S0 be a current local minimum.
3) Randomly generate N solutions around S0 within a search radius R. Store the N solutions, called neighborhood, in a set X.
4) Evaluate the objective value of each member in X via objective functions. Set S1 as a member giving the minimum cost.
5) If f(S1) < f(S0), put S0 into the TL and set S0=S1, otherwise, store S1 in the TL instead.
6) If the termination criteria: count= countmax or desired specification are met, then stop the search process. S0 is the best solution, otherwise Update count= count+1, and go back to 2.

Adaptive Tabu Search: Modified version of the tabu search. Possesses two new mechanisms, back-tracking (BT) as one of the diversification strategies and adaptive radius (AR) as one of the intensification strategies. It consist of 9 steps:

1) Initialize a search space (Ω), TL = ∅, search radius (R), count, and countmax.
2) Randomly select an initial solution S0 from a certain search space Ω. Let S0 be a current local minimum.
3) Randomly generate N solutions around S0 within a search radius R. Store the N solutions, called neighborhood, in a set X.
4) Evaluate the objective value of each member in X via objective functions. Set S1 as a member giving the minimum cost.
5) If f(S1) < f(S0), put S0 into the TL and set S0=S1, otherwise, store S1 in the TL instead.
6) Activate the BT mechanism, when a local entrapment occurs.
7) If the termination criteria: count= countmax or desired specification are met, then stop the search process. S0 is the best solution, otherwise go to 8.
8) Invoke the AR mechanism, once the search approaches the local or the global solution to refine searching accuracy.
9) Update count= count+1, and go back to 2.

Suwannarongsri and Puangdownreong run algorithms of GA, TS and ATS in MATLAB on Pentium ®, 2.00 GHz CPU, 1 GB RAM to solve ten real world TSP problems.

TSP Problems	No. of Cities	Opt. distance (km.)
Eil51	51	426
Berlin52	52	7,542
St70	70	675
Pr76	76	108,159
Eil76	76	538
Rat99	99	1,211
Rd100	100	7,910
KroA100	100	21,282
KroB100	100	22,141
Ch150	150	6,528

Each problem was tested over 20 times to calculate the average optimum distance and the average search time. The results obtained found that GA, TS and ATS can find the optimum distant of all problems, being the ATS algorithm that outperforms other two algorithms. Second is the TS and third is GA. That is because of the BT mechanism in ATS can efficiently escape such the local entrapments.

TSP problems	Opt. distance (km.)	Obtained solutions (average distance (Km.)) by AI						average search time (sec.) by AI		
		GA	%Err	TS	%Err	ATS	%Err	GA	TS	ATS
Eil51	426	441.46	3.63	445.05	4.47	438.12	2.85	2.72	2.53	2.91
Berlin52	7,542	7,833.26	3.86	8,152.79	8.10	7,702.16	2.12	3.58	2.15	4.14
St70	675	695.44	3.03	738.78	9.45	684.62	1.43	3.84	3.19	4.22
Pr76	108,159	116,273.12	7.50	123,240.57	13.94	110,478.35	2.14	4.13	4.05	4.57
Eil76	538	548.26	1.91	582.62	8.29	540.17	0.40	4.05	3.86	4.43
Rat99	1,211	1,274.84	5.27	1,295.42	6.97	1,213.50	0.21	6.28	5.83	7.35
Rd100	7,910	8,506.51	7.54	8,788.39	11.10	8,412.62	6.35	10.68	9.12	13.46
KroA100	21,282	22,875.42	7.49	23,644.18	11.10	21,525.56	1.14	11.97	10.89	15.13
KroB100	22,141	24,765.94	11.86	25,349.36	14.49	22,568.14	1.93	12.52	11.18	15.46
Ch150	6,528	6,986.35	7.02	7,012.21	7.42	6,612.74	1.30	14.84	12.57	16.73

Suwannarongsri and Puangdownreong conclude that the genetic algorithms (GA), the tabu search (TS), and the adaptive tabu search (ATS), are most popular and powerful optimization methods, however, the ATS outperforms other algorithms with most optimum solution found with reasonable time consumed.

Genetic Algorithm Code for Traveling Salesman Problem

https://github.com/germanreyga/Traveling-Salesman-GA/blob/master/travelingSalesmanSolver.py

Reference:

- Johnson, B., (2017). *Genetic Algorithms: The Travelling Salesman Problem*, Medium, Recuperado de: https://medium.com/@becmjo/genetic-algorithms-and-the-travelling-salesman-problem-d10d1daf96a1
- Stoltz E., (2018). *Evolution of a salesman: A complete genetic algorithm tutorial for Python*, Medium, Recuperado de: https://towardsdatascience.com/evolution-of-a-salesman-a-complete-genetic-algorithm-tutorial-for-python-6fe5d2b3ca35
- Suwannarongsri, Supaporn & Puangdownreong, Deacha, (2012). Solving traveling salesman problems via artificial intelligent search techniques. 137-141.