Traveling Salesman Report

Jonah TURNER
TPA11 IAFA

Metaheuristic Search

The metaheuristic search was implemented in traveling_salesperson.py in Python. I did use the external library Numpy to generate the random sequences for the initial paths. Any recent version of Numpy using Python 3.8 or later will be able to run the code.

The Python script implements steepest hill climbing and tabou list search. The script first performs one iteration of steepest hill climbing then iterates over the same algorithm \`max_essais\` times, which can be passed in as a command line argument. The steepest hill climbing algorithm uses both 2-opt and 2-swap to find neighbors, while the tabou list uses only 2-swap.

Runtime may very significantly based on the number of iterations performed, but increased iterations on the larger data files frequently produces better results. The tabou search is cheaper to perform than iterating over steepest hill climbing, so the max iteration parameter default is much higher.

ZIMPL Program

The set V represents the cities in the TSP. Edges, denoted by E, symbolize potential paths between these cities and are defined for pairs of cities where the first is less than the second. The program further utilizes a powerset P[] of V, along with an index set K, to facilitate the definition of constraints. Coordinates for each city are stored in parameters px[V] and py[V]. The decision variables x[E] are binary, indicating the presence or absence of an edge in the tour. The objective function seeks to minimize the total distance of the tour, calculated as the sum of Euclidean distances for each selected edge. Two critical constraints are imposed: two_connected, which ensures each city is connected to exactly two others, forming a continuous tour, and no_subtour, which prevents the formation of smaller, isolated tours by limiting the number of edges in any city subset. This model aims to identify the shortest tour that visits each city exactly once and returns to the starting point, adhering to the stipulated constraints to ensure a valid and complete tour solution.

The solution should have included further constraints to limit the complexity of the search like the Miller-Tucker-Zemlin (MTZ) subtler elimination strategy, but a working implementation was not achieved.

Results

The ZIMPL program was unfortunately unable to read the tsp50 data without running out of memory, so the best results were obtained with the Python solution.

Because a specific seed was not set, the results are somewhat random, especially for the instances with more cities. A well-performing run produces the following results for tsp50.txt using the Tabou search:

Cities path - [21, 23, 25, 48, 18, 19, 49, 20, 22, 24, 9, 13, 15, 16, 47, 17, 14, 12, 11, 10, 0, 33, 30, 32, 28, 26, 27, 29, 31, 34, 41, 39, 38, 37, 35, 36, 40, 43, 44, 42, 1, 3, 5, 45, 4, 46, 2, 6, 7, 8]

Distance - 473.32

ZIMPL Results

	Optimum Reached	Value Obtained	Time
Tsp5	Yes	194.04	0.0s
Tsp25	No	OOM	~
Tsp50	No	OOM	~
Tsp101	No	OOM	~

Python Metaheuristics Results (max_essais = 10, tabou_iters = 200)

	Value Obtained	Best Method	Total Time(s)
Tsp5	194.04	SHC	0.002
Tsp25	279.99	Iterated SHC	1.27
Tsp50	446.47	Tabou	10.05
Tsp101	777.822	Tabou	40.53

Reported distance for Tsp25 lower than the optimal suggested on Moodle, path found is the following (indexed at 0) :
[21 23 18 19 24 22 20 0 2 3 5 1 4 8 7 6 14 12 17 11 10 9 13 15 16]

ZIMPL Reference: https://zimpl.zib.de/download/zimpl.pdf