Proposed Heuristic for the Selective Traveling Salesman Problem

Raúl Josafat Correa Ascencio

Mathematical formulation

For each node i of G, a weight wi is given

The objective is to find a cycle Y in G such that the sum of the weights of nodes in Y is maximized

The sum of the costs cij belonging to Y must remain less than or equal to L, where L is a given budget

Constructive Heuristic

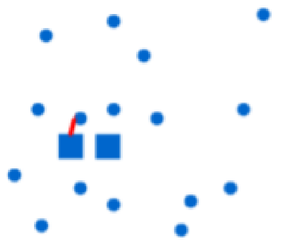

Image 1

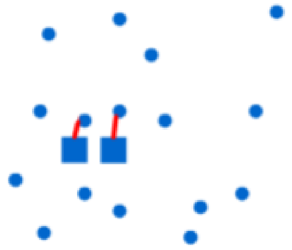

Image 2

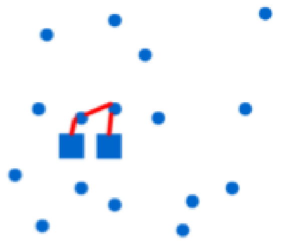

Image 3

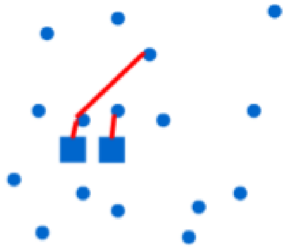

Image 4

1. Locate the *startingPoint* and the *endingPoint* and store them respectively in the lists *Beginning* and *End*
2. *pendingPlaces* = *Places* \ (*Beginning* U *End*)
3. *noPlaces* is the number of elements in *pendingPlaces*
4. *N* = 1
5. While *N* <= *noPlaces*
    1. Pick a *Place1* from *pendingPlaces* whose ratio *Profit / Eucledian(StartingPoint, Place1)* is the *N* best and add it to the end of the list *Beginning*
    2. *M* = 1
    3. While *M* <= *noPlaces*
        1. Pick a *Place2* from *pendingPlaces* whose ratio *Profit / Eucledian(Place2, endingPoint)* is the *M* best other than *Place1* and add it to the beginning of the list *End*
        2. If the cost of the path (*Beginning* + *End*) <= *timeLimit*
            1. *startingPoint* = *Place1*
            2. *endingPoint* = *Place2*
            3. Go to line 2
        3. Else
            1. *Beginning* = *Beginning* \ *Place1*
            2. *End* = *End* \ *Place2*
        4. M = M + 1
    4. N = N + 1
6. The solution is the sum of *Profit* of the places in path (*Beginning* + *End*)

Improving Heuristic - Spacing

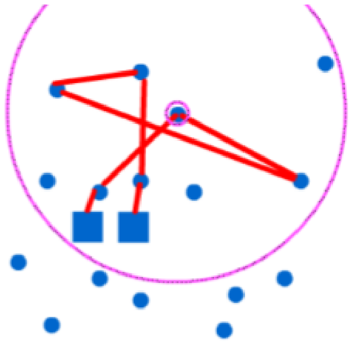

Image 1

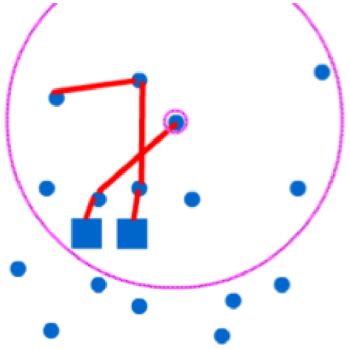

Image 2

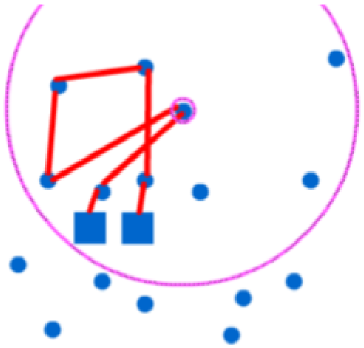

Image 3

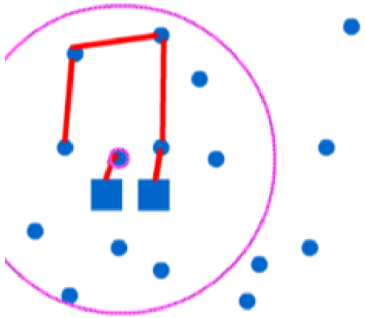

Image 4

Given a set of *Places*, a *Solution* path and its *SProfit* for the STSP:

neighborhood is a list where neighbors will be stored

1. *Radius* = 3 * (*Cost* of *Solution*) / (*noPlaces* of *Solution*)
2. *bestProfit* = *SProfit*
3. *bestSolution* = *Solution*
4. For each *Place* in *bestSolution*
    1. *unvisitedPlaces* = *Places* \ *Solution*
    2. If there is an *unvisitedPlace* whose Eucledian distance to *Place* <= *Radius*
        1. *newSolution* = change *Place* for *unvisitedPlace* in *Solution*
        2. If changing *Place* for *unvisitedPlace* results in a path whose *uCost* <= *timeLimit* and whose *uProfit* >= *SProfit*
            1. *bestSolution* = *newSolution*
            2. *bestProfit* = *uProfit*
            3. Go to line 4

Improving Heuristic - Inserting

5. *unvisitedPlaces = Places \ bestSolution*
6. If there is an *unvisitedPlace* whose Eucledian distance to *location <= Radius*
1. If adding the *unvisitedPlace* between *location* and next *location* results in a path whose *cost <= timeLimit*
2. If the *newProfit* of this *newSolution >= bestProfit*
    1. *bestSolution = newSolution*
    2. *bestProfit = newProfit*
3. The improved solution is the *bestProfit*

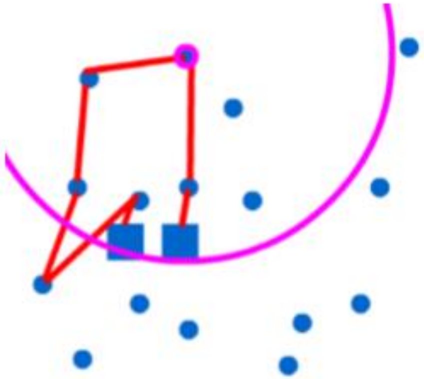

Image 1

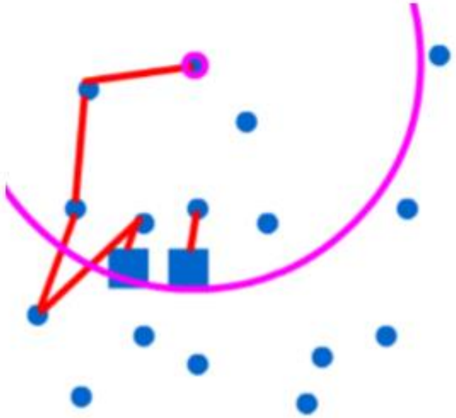

Image 2

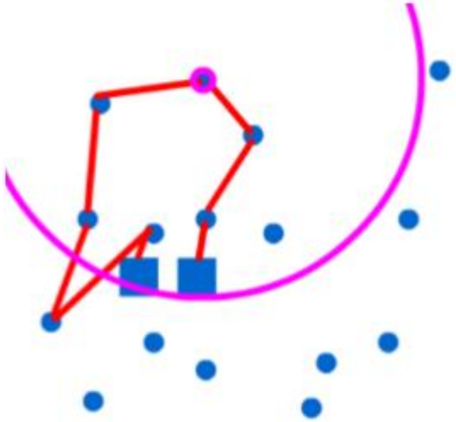

Image 3

GRASP

- Instead of picking the node with the best profit-to-cost ratio, the first **k best** options were scrambled.
- Restricted Candidate List dependent of alpha value:
- RCL = (1 - α) * maxRatio + α * minRatio
- All alphas start with the same probability of being chosen
- Probability updated each 100 iterations
- $P_i = \frac{\frac{\hat{Z}_i}{Ƶ}}{\frac{\sum \hat{Z}}{Ƶ}}$; P_i is the probability of α= i, $\hat{Z}_i$ is the average profit obtained with α= i, $\sum \hat{Z}$ is the sum of all the average profits, and Ƶ is the maximum profit obtained up to that point
- 1000 iterations done with this random heuristic

GRASP Analysis

Algorithm evaluated using Tsigilirides set 1, 2, 3, 64, and 66 and compared against Chao's best results.

K	Average rise from Local Search (%)	Average deviation from Chao's (%)	Best known solutions	Mode of α from improved solutions	Total time (s)	Average time(s)
2	2.15	-13.26	12	0.5	6404.46	71.96
3	**9.94**	**-5.98**	**15**	**0.8**	**6247.98**	**70.20**
4	9.06	-6.91	13	0.8	6262.04	70.36
5	8.93	-7.13	13	0.5	6083.75	68.36
Avrg	7.52	-8.32			6249.5575	70.22
				Total	24998.23	

Set 64 comparison with Chao's Solution

	K-best (K = 3) & α (best result out of 1000)		Chao		
Budget	Profit	Time (s)	Profit	Time (s)	% deviation
15	78	78.498	96	13.01	-18.75
20	**294**	**57.345**	**294**	27.86	**0.00**
25	**390**	**66.946**	**390**	238.9	**0.00**
30	468	75.264	474	74.48	-1.27
35	528	90.466	570	139.86	-7.37
40	666	104.882	714	137.9	-6.72
45	780	118.269	816	204.98	-4.41
50	852	121.526	900	231.57	-5.33
55	942	127.659	984	246.18	-4.27
60	1026	133.688	1044	264.77	-1.72
65	1080	134.156	1116	232.57	-3.23
70	1140	136.927	1176	230.95	-3.06
75	1176	137.636	1224	223.12	-3.92
80	1212	138.792	1272	212.27	-4.72

Average Time		**Average Deviation**
Proposed Heuristic	Chao's Heuristic	
70.20	82.788	-4.63

Conclusions

Having **two paths** may lead to them going in opposite directions if it is too greedy, but by randomizing the selection in both paths increases the variation when doing the 1000 iterations, augmenting the chance of getting a new solution which may be better.

Further improvement may involve evaluation the angle between both paths to avoid selecting in the second path places that are too far apart from the selected in the first path.

Using a **reactive GRASP** lead to some good results and help the user not to worry about checking alphas which may not work so well, providing time for checking more iterations and attempring to improve further the solutions.

Bibliography

Chao, I-M., Golden, B., & Wasil, E. (1995). A fast and effective heuristic for the orienteering problem. EUROPEAN JOURNAL OF OPERATIONAL RESEARCH, 475-489.

Gutin, G., & Punnen, A. (2006). The Traveling Salesman Problem and Its Variations. New York: Springer.