George Hatzis-Schoch

5/15/17

John Anderson

Final Project

A Model for the Traveling Salesperson Problem

Introduction

The aim for this project was to create an ACT-R model which simulates a human solving the Euclidean Traveling Salesperson Problem (Euclidean TSP). The classic TSP is defined as follows. In a given graph, find the shortest complete tour through the vertices (cities), starting and ending with the same city. The Euclidean TSP problem is the same problem, except the cities are a set of points in a plane, where the Euclidean distance between any two points is the "cost" to traverse from one city to the other. The goal is to minimize the total cost of the tour.

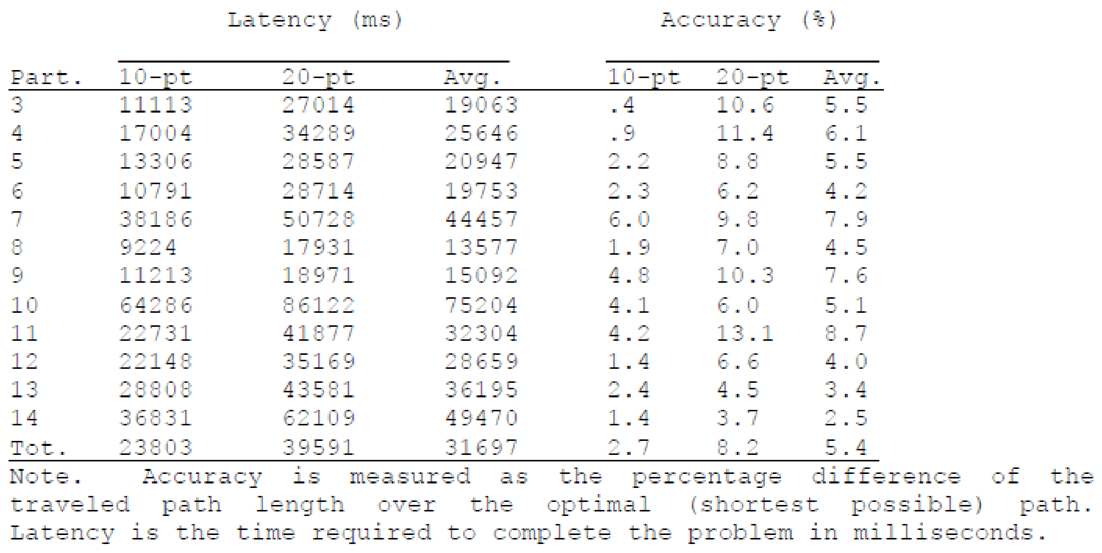
The Euclidean TSP is an interesting modeling task due to the intuitive nature of the task along with its intrinsic difficulty. In fact, the Euclidean TSP problem is an NP-hard problem. This means that no polynomial-time algorithm is currently known which solves the Euclidean TSP, and so providing optimal solutions to the problem is inherently difficult. However, when it comes to approximating the TSP problem, there are many fairly accurate heuristic algorithms. Researchers have used simulations of ant colonies and other clever randomized algorithms to obtain reliable approximations of the TSP. It has been found that humans are both accurate and asymptotically efficient at approximating the Euclidean TSP. This project is an attempt to model the performance of humans on the Euclidean TSP as documented in Bradley J. Best's dissertation (Best 2004).

Experiment

The experiment that this project targets to model is Experiment 1 in (Best 2004). In this experiment, participants were asked to find a tour of a graph presented on a computer program. The computer program was running on a laptop with a display of resolution 800 by 600 pixels. The graphs used in this experiment were the same as the graphs used in (MacGregor and Ormerod 1996), with the coordinates scaled by a factor of 2 to better fit the computer screen. Each graph is constructed with an exterior rotated polygon with varying amounts of interior points. Participants were asked to complete a tour of the graph presented on the computer program. To visit a node, they used a computer mouse to click on the nodes in the program. The first node visited for each problem was free for the participant to choose.

Each problem in the original experiment started by prompting the subject with dialogue asking if they were ready to begin. To begin, the participant clicked an "OK" button which was at the center of the screen. This re-centered the mouse at the beginning of each trial. For a trial to end, the participant was required to complete a tour of the graph. Participants could complete an unfinished tour. However, none of the participants in Experiment 1 attempted to do this. The experiment collected the accuracy of the tours produced by the participants and the latency of the participants on each graph. Below is a table summarizing the results:

Figure 1- Latency and Accuracy data from experiment 1 (Best 2004). Target row is "Tot.".



	Latency (ms)			Accuracy (%)		
Part.	10-pt	20-pt	Avg.	10-pt	20-pt	Avg.
3	11113	27014	19063	.4	10.6	5.5
4	17004	34289	25646	.9	11.4	6.1
5	13306	28587	20947	2.2	8.8	5.5
6	10791	28714	19753	2.3	6.2	4.2
7	38186	50728	44457	6.0	9.8	7.9
8	9224	17931	13577	1.9	7.0	4.5
9	11213	18971	15092	4.8	10.3	7.6
10	64286	86122	75204	4.1	6.0	5.1
11	22731	41877	32304	4.2	13.1	8.7
12	22148	35169	28659	1.4	6.6	4.0
13	28808	43581	36195	2.4	4.5	3.4
14	36831	62109	49470	1.4	3.7	2.5
Tot.	23803	39591	31697	2.7	8.2	5.4

Note. Accuracy is measured as the percentage difference of the traveled path length over the optimal (shortest possible) path. Latency is the time required to complete the problem in milliseconds.

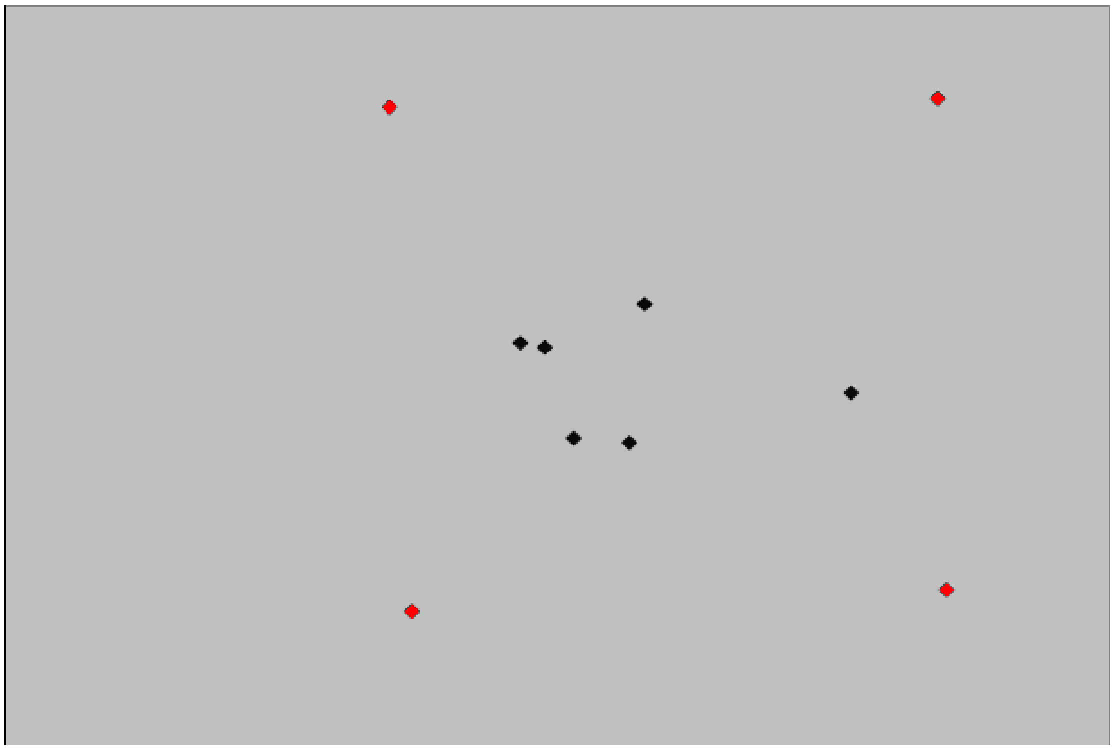
There are two main features of the data which I will focus on in modeling. First, the overall accuracy of the participants is very high. In 10 point graphs, the participants had 2.7% error on average, and 8.2% average error in 20 point graphs. Remarkably, the latency of the participants shows a linear relationship between the time to complete a tour and the number of cities in the problem. 20 point problems took approximately twice the time of 10 point tasks. With such high accuracy and a linear relationship between latency and problem size, these results indeed support the effectiveness of humans in approximating the TSP, and are the two main features I will target.

Modeling Techniques

Since a computer program was used to do this experiment, an ACT-R model naturally fits this experiment. To model the graphs, experimental window of size 800 by 600 is used. At the start of a trial, the model is presented with a “Start” button at the center of the screen. By clicking the button, a graph is presented from the current set of problems, and this denotes the start of the trial. The model must click a button to add the corresponding vertex of the graph to the tour. Once a button has been clicked, an edge is drawn from the last visited city on the tour to the clicked button. A trial is completed once the tour visits all vertices in the graph. The graphs presented in the experiment are scaled by a factor of 2 from the original coordinates used in (MacGregor and Ormerod 1996). In one run of the experiment, there are 14 total trials the model will complete, each corresponding to a different graph.

Information about the convex hull of the graph is theorized to be readily available in people’s visual systems. Thus, for the purposes of this model the points (buttons) lying on the convex hull of the graph are colored red initially before each trial begins. The model has access to this information by querying the visual-location buffer for visual objects which are red in the experimental window. The convex hull of each graph is pre-computed and fed to the model with the coordinate points as a corresponding boolean value for each city on the graph (I.e. true if the city is on the convex hull of the grap, and false otherwise). Below is an example of an initial graph in the experimental window.

Figure 2- Illustration of the initial experimental window for the first 10-point graph in Experiment 1 (Best 2004).



Each button in the experimental window has an action function **button-action**, which does the following when called (in order):

1. If **current** is not **nil**, a line is added to the experimental window from **current** to the button corresponding to the **button-action** function which was called.
2. Sets a variable **current** to store the value of the current clicked button.
3. Labels 3 points yellow in the experimental window, which are the points the model will consider to visit next.

At the start of a trial, the model picks any city that is on the convex hull to begin its tour. This is done by making a request to the visual-location buffer for a red button. When a button is clicked, the action function is called. In this function, **current** is set and the three points which are colored yellow by the button action function are the following:

(1) The next unvisited city on the convex hull in the counter-clockwise direction.

(2) The closest unvisited city to the line connecting **current** and the point found in (1).

(3) The closest unvisited city to **current** that is not on the convex hull and not the point found in (2).

External lisp code is executed which finds each of the three points. Point (1) is found by a linear search on the convex hull points of the graph. Point (2) is found using several geometric formulas to compute the perpendicular distance from each point in the graph from the line connecting **current** and the point found in (1), and taking the point corresponding to the minimum distance found. Similarly, point (3) is found by calculating the distance to **current** from each point not on the convex hull in the graph and not the point in (2), and taking point corresponding to the minimum distance found.

Once all three buttons are found by the action function, they are colored yellow. For the model to choose from the three identified points, it makes a request to the visual location buffer for the nearest yellow point to the current attended location (which coincides with **current**). Once the model finds the closest yellow point, it can then click on the chosen button and the process to find three new yellow points restarts.

The trial terminates when the tour is completed. In the original experiment in (Best 2004), it was up to the participant to check if the tour was complete, and therefore it was possible for tours to be ended prematurely in error. However, based off the data collected in experiment 1, no participants made this error, and so the model does not either. The model can begin the process to end a trial once it is not able to find all three described points. It can then complete the tour by visiting the initial city located on the convex hull one last time. The location of the initial city is kept track of by an external variable, and the model is forced to revisit it when no other un-visited cities remain.

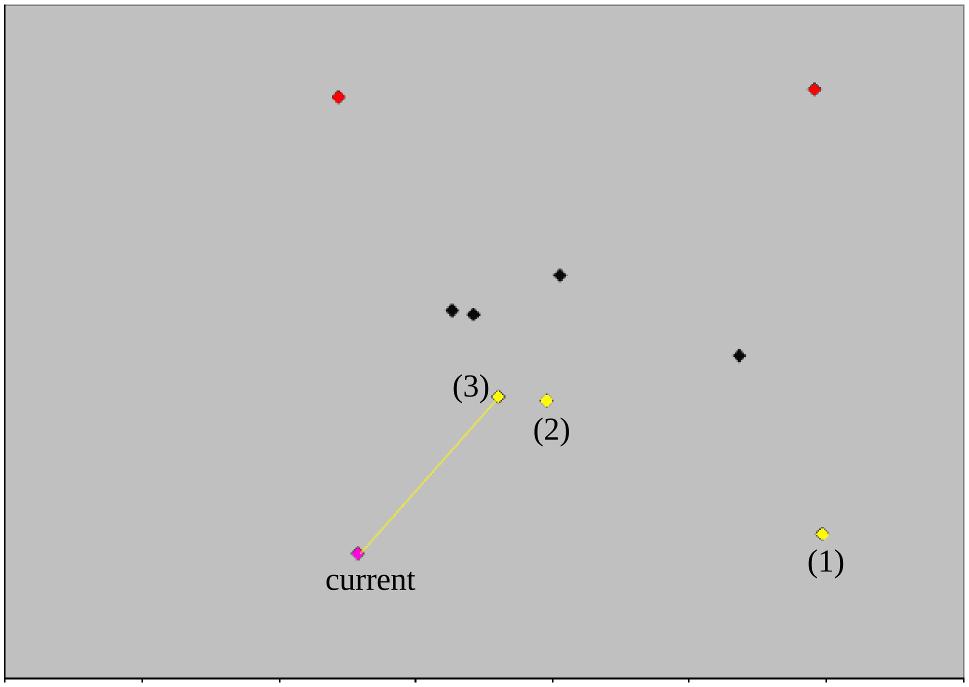


Figure 3- Illustration of the 3 "special" points colored by external lisp code, with the model choosing the closest one to "current".

Results

Latency (ms)			Accuracy (%)		
10 pt	20 pt	Avg.	10 pt	20 pt	Avg.
13648	24635	19142	15.0	16.4	15.7

Figure 4- Averaged results of 3 runs of the experiment before changing Fitt's law parameter.

Above are the averaged results from 3 separate runs of the experiment with the initial model. In terms of latency, the model is far too fast. It performs roughly twice as fast than the human performance recorded in (Best 2004). However, there is still some hope, as the model does roughly have a linear relationship between number of cities and latency, with 20 point problems taking about twice the time as 10 point problems. The accuracy of the model has a 12% difference from human performance for 10 point cities, 8% difference for 20 point cities.

This discrepancy in performance of the model can be attributed to the form of the graphs. Once the model reaches the interior of the convex hull, it generally picks up nearly all the points of the interior. Since these points are closer together, any deviances from the optimal path made are less costly here. Thus, since the 20 point graphs have more points on the interior, the model generally generates better paths. In addition, the model’s choice of when to enter or leave the interior leads to very different results. Since the model is only making local decisions, if it chooses a poor entry or exit point for the interior, the accuracy percentage for the model can spike. Furthermore, the model does not plan its closing of the tour. When all the cities of the graph are visited, the model will simply choose to visit the initial city one last time. However, this can add a large cost to the tour depending on how far away the model’s last visited city is from the initial city.

To fix the accuracy of the model, a new strategy for the model would be required. Aside from the use of the pre-computed convex hull of the graph, the current strategy is entirely local. The external lisp code does not consider the overall features of the graph, but rather only labels points based off the most recently visited point and the convex-hull point. Due to this, the model can often make some poor decisions which most of the participants likely did not make. One way the global features of the graph could be considered is for the model to choose from the 3 yellow points using some other method, different than the greedy distance method. For example, choosing the nearest point such that when this point is visited, the edge drawn does not form a "cross"[1] in the path could improve performance.

To fix the latency of the model, there are two potential causes to consider. The first is that the external lisp code is accounting for too much of the cognitive work which a participant would undergo in producing a tour. This results in the model spending too little time "thinking". I believe this is certainly a source of latency error, and fixing this would require a complete rework of the model's method for choosing points. Another cause for the low latency could be the model's movement of the mouse. The model makes no mistakes in clicking buttons, and thus progresses along the path at a very fast rate. This could account for some of the latency discrepancy. To fix this, I tried setting the parameter ":mouse-fitts-coeff" to the value 0.2 (this is default set to 0.1), which due to Fitts' law causes the model's mouse movement to slow down. The results in the data due to this change are below:

Latency (ms)			Accuracy (%)		
10 pt	20 pt	Avg.	10 pt	20 pt	Avg.
17587	30606	24097	9.9	13.4	11.6

Figure 5- Average results of experiment after changing Fitt's law parameter.

The latency of the model increased dramatically due to this change, and it much better fits the target data. This is supported by the possibility that most of the latency in this experiment is accounted for by mouse movement. Furthermore, the linear relationship between latency and number of cities is still present. Another aspect the data highlights is how variable the model's performance is. There is a 6% error decrease in the 10 point graphs from the data collected before the change and after the change, and a 3% decrease in 20 point graph accuracy. This error

[1] A "cross" in a tour is never optimal.

decrease is not due to the parameter change, but rather just due to the inherent unpredictability of the model’s current method due to the arbitrary choice of the starting point on the hull.

Conclusion

Overall, the model was not fully successful at modeling human performance in this experiment. Among the model’s successes are displaying a linear relationship between latency and number of cities, and its fairly accurate latency results (post Fitts’ law parameter change). Since the linear relationship is a critical feature of the target data, this shows that the model is on the right track. However, the model is still a bit fast in its performance. One way to improve this would be to implement mis-clicking in the model. This would help increase the latency, but would require data on participant clicking accuracy and additional correction mechanisms would have to be added to the model.

In terms of tour accuracy, the current strategy of the model has several problems. One issue is that the model frequently makes crosses in its path. Preventing crosses from occurring would be the logical starting point in improving this model in future attempts at modeling this experiment. Similar to how the three “yellow” points are colored, this could be done using external lisp code or perhaps even done by the model itself. Another issue with the model is that aside from the convex hull of the graph, local features of the graph are mostly considered. This lack of global features may be necessary in the method to allow for the linear relationship to be achieved in the model. However, since the model still has some extra latency to make up for, considering one or two global features in the graph such as choosing a better starting point could help decrease the accuracy percentages and increase the overall latency by a small amount. To conclude, while this project was not a success on all ends, it certainly helped reveal some of the areas of difficulty in modeling the Euclidean TSP, and provided insight into where improvements could be made in further modeling of the experiment.

Appendix

Graphs coordinates, (not scaled by a factor of 2):

Format:

# cities total/#cities interior

Optimal cost

Coordinates

10/6
528.51
196.83 146.3
81.42 144.49
85.83 31.13
198.74 35.69
114.25 90.37
178.62 79.78
135.33 99.61
120.41 69.85
109.31 91.33
132.41 69.11

10/5
558.15
215.06 117.67
134.03 169.78
62.38 109.39
89.26 28.15
192.35 29.51
128.76 55.87
139.44 90.92
142.22 71.20
152.24 62.73
168.57 103.29

10/4
595.30
219.71 83.18
177.94 160.43
97.40 157.72
60.06 86.99
104.47 18.32
175.66 18.39
155.34 58.40
150.41 126.27
146.55 87.41
127.32 93.50

10/3
559.55
206.66 134.24
137.58 169.96

77.94 140.48
61.22 76.07
111.63 15.20
170.66 16.11
217.29 69.34
168.6 98.57
162.65 81.02
160.42 107.24

10/2
566.22
219.95 92.83
194.97 148.29
143.04 169.94
80.85 143.86
60.00 90.03
79.34 37.85
137.32 10.04
194.27 31.23
171.90 108.66
103.42 93.35

10/1
530.89
219.96 92.44
197.63 145.49
147.71 169.63
96.68 157.26
65.90 120.16
65.18 61.69
105.65 17.75
157.03 11.83
197.15 34.02
177.33 89.20

10/-1
758.66
220 170
60 170
60 10
220 10
220 90
180 130
100 50
140 90
180 90
200 90

20/16
593.81
200 142
80 141
84 34
194 32
176 94
157 100

168 124
143 119
127 123
125 120
109 118
109 101
102 85
122 82
129 72
129 59
153 44
147 73
158 80
173 83

20/14
663.61
210 128
138 169
73 132
69 55
145 11
209 51
157 94
176 121
153 137
138 103
135 105
132 98
117 94
112 80
116 63
129 43
145 55
158 46
155 76
178 80

20/12
688.33
213 122
170 164
112 164
66 120
64 66
110 16
167 15
212 56
152 93
156 115
145 111
139 106
124 109
121 91
126 87
131 82

130 45
149 67
149 76
185 82

20/10
698.83
214 117
191 151
136 169
94 154
64 113
65 63
94 26
146 11
185 25
216 67
159 94
153 106
136 129
123 103
108 97
114 85
129 81
132 46
157 70
156 84

20/8
725.31
215 115
192 149
164 166
122 167
82 144
64 112
62 76
84 34
114 15
166 15
195 33
218 74
173 111
148 105
122 113
99 92
130 88
125 59
147 56
167 82

20/6
703.89
217 109
199 143
168 164

146 169
109 163
76 137
62 105
62 77
77 42
107 18
139 11
172 17
198 35
217 69
182 114
142 111
112 97
114 82
147 47
150 87

20/4
703.81
217 111
210 128
187 154
148 169
131 169
97 157
76 137
62 105
61 80
74 46
95 25
125 12
155 12
178 20
209 50
219 74
182 98
127 111
133 42
161 63

Works Cited

Best, Bradley J. "Modeling Human Performance on the Traveling Salesperson Problem: Empirical Studies and Computational Simulations." Carnegie Mellon University, 2004. Web. 13 May 2017.

Macgregor, J. N. and T. Ormerod. "Human performance on the traveling salesman problem." *Perception & Psychophysics* (1996). Web.