UNIVERSITY OF CAMBRIDGE

Exactly solving TSP using the Simplex algorithm

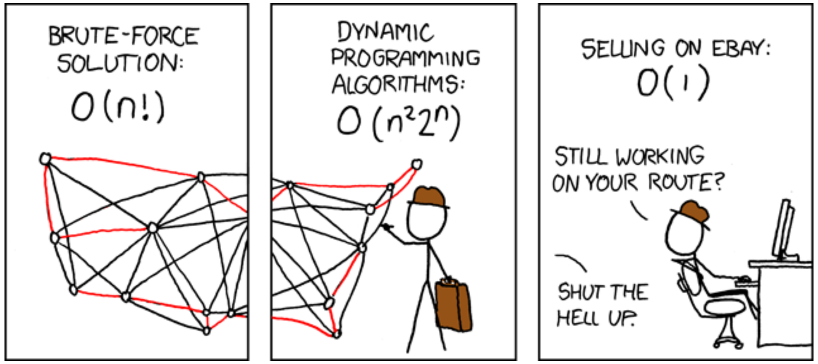
Andrej Ivašković, Thomas Sauerwald

CST Part II
ADVANCED ALGORITHMS

15 May 2019

(original slides by Petar Veličković)

Travelling Salesman Problem (http://xkcd.com/399/)



Aside: Held–Karp algorithm

- Use a *dynamic programming* approach. *Main idea:* solve the slightly simpler problem of the shortest *path* visiting all nodes, then route the end to the beginning.
- Assume (wlog) that the path starts from node 1. Given a node x and set of nodes S with $1 \in S$, maintain the solution $dp(x, S)$ as the shortest path length starting from 1, visiting all nodes in S, and ending in x.
- Base case: $dp\left(1, \{1\}\right) = 0$.
- Recurrence relation:

$$dp(x, S) = \begin{cases} \min\limits_{y \in S} \left\{ dp\left(y, S \setminus \{x\}\right) + c_{yx} \right\} & x \in S \wedge 1 \in S \\ +\infty & \text{otherwise} \end{cases}$$

Aside: Held–Karp algorithm

- Finally, $dp(x, V)$ will give the shortest path visiting all nodes, starting in 1 and ending in x.
- Now the optimum TSP length is simply:

$$\min_{x \in V} \left\{ dp(x, V) + c_{x1} \right\}$$

  The cycle itself can be extracted by backtracking.
- The set S can be efficiently maintained as an n-bit number, with the i-th bit indicating whether or not the i-th node is in S.
- Complexity: $O(n^2 2^n)$ time, $O(n2^n)$ space.

LP formulation

- We will be using *indicator variables* x_{ij}, which should be set to 1 if the edge $i \leftrightarrow j$ is included in the optimum cycle, and 0 otherwise. To avoid duplication, we impose $i > j$.
- An adequate linear program is as follows:

$$
\begin{array}{lrcl}
\text{minimise} & \sum_{i=1}^{n} \sum_{j=1}^{i-1} c_{ij} x_{ij} & & \\
\text{subject to} & & & \\
\forall i.\ 1 \leq i \leq n & \sum_{j<i} x_{ij} + \sum_{j>i} x_{ji} & = & 2 \\
\forall i, j.\ 1 \leq j < i \leq n & x_{ij} & \leq & 1 \\
\forall i, j.\ 1 \leq j < i \leq n & x_{ij} & \geq & 0
\end{array}
$$

- This is *intentionally* an incompletely specified problem:
  - We allow for *subcycles* in the returned path.
  - We allow for "partially used edges" ($0 < x_{ij} < 1$) – this LP approximates an integer program.

LP solution

- If the Simplex algorithm finds a correct cycle (with no subcycles or partially used edges) on the underspecified LP instance, then we have successfully solved the problem!
- Otherwise, we need to resort to further specifying the problem by adding additional constraints (manually or automatically).

Further constraints: subcycles

- If the returned solution contains a subcycle, we may eliminate it by adding an explicit constraint against it, and then attempt solving the LP again.
- For a subcycle containing nodes from a set $S \subset V$, we may demand at least two edges between S and $V \setminus S$:

$$\sum_{\substack{i \in S \\ j \in V \setminus S}} x_{\max(i,j),\min(i,j)} \geq 2$$

- We will not add all of these contraints – why?
- We often don't need to add all the constraints in order to reach a valid solution.

Further constraints: partially used edges

- If the returned solution contains a partially used edge, we may attempt a *branch&bound* strategy on it.
- For a partially used edge $a \leftrightarrow b$, we initially add a constraint $x_{ab} = 1$, and continue solving the LP.
- Once a valid solution has been found, we remove all the constraints added since then, add a new constraint $x_{ab} = 0$, and solve the LP again.
- We may stop searching a branch if we reach a worse objective value than the best valid solution found so far.
- The optimum solution is the better out of the two obtained solutions! If we choose the edges wisely, we may often obtain a valid solution in a complexity much better than exponential.

Demo: abstract

SOLUTION OF A LARGE-SCALE TRAVELING-SALESMAN PROBLEM*

G. DANTZIG, R. FULKERSON, AND S. JOHNSON
The Rand Corporation, Santa Monica, California

(Received August 9, 1954)

It is shown that a certain tour of 49 cities, one in each of the 48 states and Washington, D. C., has the shortest road distance.

THE TRAVELING-SALESMAN PROBLEM might be described as follows: Find the shortest route (tour) for a salesman starting from a given city, visiting each of a specified group of cities, and then returning to the original point of departure. More generally, given an n by n symmetric matrix $D=(d_{IJ})$, where d_{IJ} represents the 'distance' from I to J, arrange the points in a cyclic order in such a way that the sum of the d_{IJ} between consecutive points is minimal. Since there are only a finite number of possibilities (at most $\frac{1}{2}(n-1)!$) to consider, the problem is to devise a method of picking out the optimal arrangement which is reasonably efficient for fairly large values of n. Although algorithms have been devised for problems of similar nature, e.g., the optimal assignment problem,[3,7,8] little is known about the traveling-salesman problem. We do not claim that this note alters the situation very much; what we shall do is outline a way of approaching the problem that sometimes, at least, enables one to find an optimal path and prove it so. In particular, it will be shown that a certain arrangement of 49 cities, one in each of the 48 states and Washington, D. C., is best, the d_{IJ} used representing road distances as taken from an atlas.

Demo: nodes

Now we will make advantage of these techniques to solve the TSP problem for 42 cities in the USA—using the *Held-Karp* algorithm would require ~ 4 hours (and unreasonable amounts of memory)!

1. Manchester, N. H.
2. Montpelier, Vt.
3. Detroit, Mich.
4. Cleveland, Ohio
5. Charleston, W. Va.
6. Louisville, Ky.
7. Indianapolis, Ind.
8. Chicago, Ill.
9. Milwaukee, Wis.
10. Minneapolis, Minn.
11. Pierre, S. D.
12. Bismarck, N. D.
13. Helena, Mont.
14. Seattle, Wash.
15. Portland, Ore.
16. Boise, Idaho
17. Salt Lake City, Utah
18. Carson City, Nev.
19. Los Angeles, Calif.
20. Phoenix, Ariz.
21. Santa Fe, N. M.
22. Denver, Colo.
23. Cheyenne, Wyo.
24. Omaha, Neb.
25. Des Moines, Iowa
26. Kansas City, Mo.
27. Topeka, Kans.
28. Oklahoma City, Okla.
29. Dallas, Tex.
30. Little Rock, Ark.
31. Memphis, Tenn.
32. Jackson, Miss.
33. New Orleans, La.
34. Birmingham, Ala.
35. Atlanta, Ga.
36. Jacksonville, Fla.
37. Columbia, S. C.
38. Raleigh, N. C.
39. Richmond, Va.
40. Washington, D. C.
41. Boston, Mass.
42. Portland, Me.

A. Baltimore, Md.
B. Wilmington, Del.
C. Philadelphia, Penn.
D. Newark, N. J.
E. New York, N. Y.
F. Hartford, Conn.
G. Providence, R. I.

Demo: adjacency matrix

TABLE I

ROAD DISTANCES BETWEEN CITIES IN ADJUSTED UNITS

The figures in the table are mileages between the two specified numbered cities, less 11, divided by 17, and rounded to the nearest integer.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41
2	8																																								
3	39	45																																							
4	37	47	9																																						
5	50	49	21	15																																					
6	61	62	21	20	17																																				
7	58	60	16	17	18	6																																			
8	59	60	15	20	26	17	10																																		
9	62	66	20	25	31	22	15	5																																	
10	81	81	40	44	50	41	35	24	20																																
11	103	107	62	67	72	63	57	46	41	23																															
12	108	117	66	71	77	68	61	51	46	26	11																														
13	145	149	104	108	114	106	99	88	84	63	49	40																													
14	181	185	140	144	150	142	135	124	120	99	85	76	35																												
15	187	191	146	150	156	142	137	130	125	105	90	81	41	10																											
16	161	170	120	124	130	115	110	104	105	90	72	64	34	31	27																										
17	142	146	101	104	111	97	91	85	86	75	51	59	29	53	48	21																									
18	174	178	133	138	143	129	123	117	118	107	83	84	54	46	35	26	31																								
19	185	186	142	143	140	130	126	124	128	118	93	101	72	69	58	58	43	26																							
20	164	165	120	123	124	106	106	105	110	104	86	97	71	93	82	62	42	45	22																						
21	137	139	94	96	94	80	78	77	84	77	56	64	65	90	87	58	36	68	50	30																					
22	117	122	77	80	83	68	62	60	61	50	34	42	49	82	77	60	30	62	70	49	21																				
23	114	118	73	78	84	69	63	57	59	48	28	36	43	77	72	45	27	59	69	55	27	5																			
24	85	89	44	48	53	41	34	28	29	22	23	35	69	105	102	74	56	88	99	81	54	32	29																		
25	77	80	36	40	46	34	27	19	21	14	29	40	77	114	111	84	64	96	107	87	60	40	37	8																	
26	87	89	44	46	46	30	28	29	32	27	36	47	78	116	112	84	66	98	95	75	47	36	39	12	11																
27	91	93	48	50	48	34	32	33	36	30	34	45	77	115	110	83	63	97	91	72	44	32	36	9	15	3															
28	105	106	62	63	64	47	46	49	54	48	46	59	85	119	115	88	66	98	79	59	31	36	42	28	33	21	20														
29	111	113	69	71	66	51	53	56	61	57	59	71	96	130	126	98	75	98	85	62	38	47	53	39	42	29	30	12													
30	91	92	50	51	46	30	34	38	43	49	60	71	103	141	136	109	90	115	99	81	53	61	62	36	34	24	28	20	20												
31	83	85	42	43	38	22	26	32	36	51	63	75	106	142	140	112	93	126	108	88	60	64	66	39	36	27	31	28	28	8											
32	89	91	55	55	50	34	39	44	49	63	76	87	120	155	150	123	100	123	109	86	62	71	78	52	49	39	44	35	24	15	12										
33	95	97	64	63	56	42	49	56	60	75	86	97	126	160	155	128	104	128	113	90	67	76	82	62	59	49	53	40	29	25	23	11									
34	74	81	44	43	35	23	30	39	44	62	78	89	121	159	155	127	108	136	124	101	75	79	81	54	50	42	46	43	39	23	14	14	21								
35	67	69	42	41	31	25	32	41	46	64	83	90	130	164	160	133	114	146	134	111	85	84	86	59	52	47	51	53	49	32	24	24	30	9							
36	74	76	61	60	42	44	51	60	66	83	102	110	147	185	179	155	133	159	146	122	98	105	107	79	71	66	70	70	60	48	40	36	33	25	18						
37	57	59	46	41	25	30	36	47	52	71	93	98	136	172	172	148	126	158	147	124	121	97	99	71	65	59	63	67	62	46	38	37	43	23	13	17					
38	45	46	41	34	20	34	38	48	53	73	96	99	137	176	178	151	131	163	159	135	108	102	103	73	67	64	69	75	72	54	46	49	54	34	24	29	12				
39	35	37	35	26	18	34	36	46	51	70	93	97	134	171	176	151	129	161	163	139	118	102	101	71	65	65	70	84	78	58	50	56	62	41	32	38	21	9			
40	29	33	30	21	18	35	33	40	45	65	87	91	117	166	171	144	125	157	156	139	113	95	97	67	60	62	67	79	82	62	53	59	66	45	38	45	27	15	6		
41	3	11	41	37	47	57	55	58	63	83	105	109	147	186	188	164	144	176	182	161	134	119	116	86	78	84	88	101	108	88	80	86	92	71	64	71	54	41	32	25	
42	5	12	55	41	53	64	61	61	66	84	111	113	150	186	192	166	147	180	188	167	140	124	119	90	87	90	94	107	114	77	86	92	98	80	74	77	60	48	38	32	6

Demo: final solution

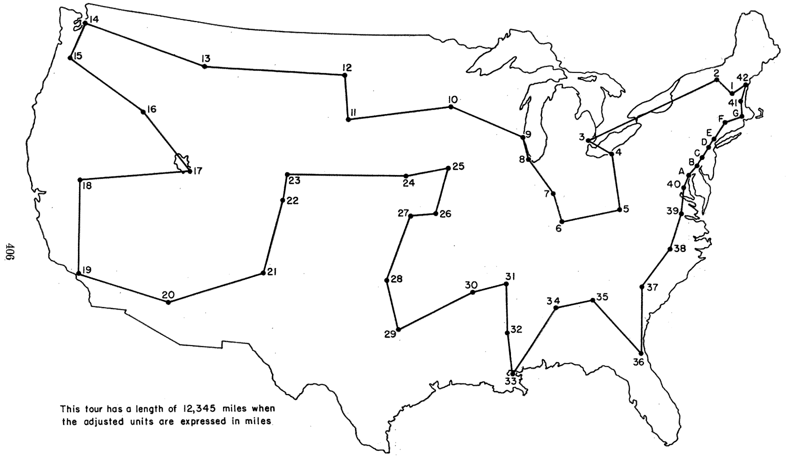


FIG. 16. The optimal tour of 49 cities.

Demo: materials

- The full implementation of this TSP solver in C++ (along with all the necessary files to perform this demo) may be found at:
  `https://github.com/PetarV-/Simplex-TSP-Solver`

- Methods similar to these have been successfully applied for solving far larger TSP instances. For example:
  `http://www.math.uwaterloo.ca/tsp/`