Randomised Algorithms

Lecture 8: Solving a TSP Instance using Linear Programming

Thomas Sauerwald (tms41@cam.ac.uk)

Lent 2023

UNIVERSITY OF CAMBRIDGE

Outline

Introduction

Examples of TSP Instances

Demonstration

The Traveling Salesman Problem (TSP)

Given a set of cities along with the cost of travel between them, find the cheapest route visiting all cities and returning to your starting point.

The Traveling Salesman Problem (TSP)

Given a set of cities along with the cost of travel between them, find the cheapest route visiting all cities and returning to your starting point.

Formal Definition

The Traveling Salesman Problem (TSP)

Given a set of cities along with the cost of travel between them, find the cheapest route visiting all cities and returning to your starting point.

Formal Definition

- Given: A complete undirected graph $G = (V, E)$ with nonnegative integer cost $c(u, v)$ for each edge $(u, v) \in E$

The Traveling Salesman Problem (TSP)

Given a set of cities along with the cost of travel between them, find the cheapest route visiting all cities and returning to your starting point.

Formal Definition

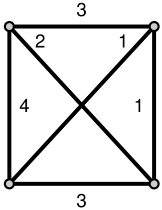
- Given: A complete undirected graph $G = (V, E)$ with nonnegative integer cost $c(u, v)$ for each edge $(u, v) \in E$
- Goal: Find a hamiltonian cycle of G with minimum cost.

The Traveling Salesman Problem (TSP)

Given a set of cities along with the cost of travel between them, find the cheapest route visiting all cities and returning to your starting point.

Formal Definition

- Given: A complete undirected graph $G = (V, E)$ with nonnegative integer cost $c(u, v)$ for each edge $(u, v) \in E$
- Goal: Find a hamiltonian cycle of G with minimum cost.

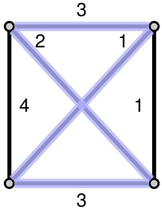


The Traveling Salesman Problem (TSP)

Given a set of cities along with the cost of travel between them, find the cheapest route visiting all cities and returning to your starting point.

Formal Definition

- Given: A complete undirected graph $G = (V, E)$ with nonnegative integer cost $c(u, v)$ for each edge $(u, v) \in E$
- Goal: Find a hamiltonian cycle of G with minimum cost.



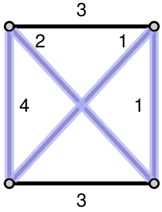
$3 + 2 + 1 + 3 = 9$

The Traveling Salesman Problem (TSP)

Given a set of cities along with the cost of travel between them, find the cheapest route visiting all cities and returning to your starting point.

Formal Definition

- Given: A complete undirected graph $G = (V, E)$ with nonnegative integer cost $c(u, v)$ for each edge $(u, v) \in E$
- Goal: Find a hamiltonian cycle of G with minimum cost.



$2 + 4 + 1 + 1 = 8$

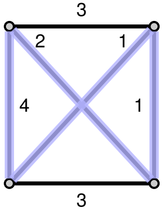
The Traveling Salesman Problem (TSP)

Given a set of cities along with the cost of travel between them, find the cheapest route visiting all cities and returning to your starting point.

Formal Definition

- Given: A complete undirected graph $G = (V, E)$ with nonnegative integer cost $c(u, v)$ for each edge $(u, v) \in E$
- Goal: Find a hamiltonian cycle of G with minimum cost.

Solution space consists of at most $n!$ possible tours!



$2 + 4 + 1 + 1 = 8$

The Traveling Salesman Problem (TSP)

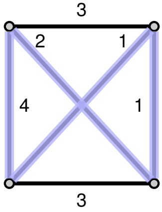
Given a set of cities along with the cost of travel between them, find the cheapest route visiting all cities and returning to your starting point.

Formal Definition

- Given: A complete undirected graph $G = (V, E)$ with nonnegative integer cost $c(u, v)$ for each edge $(u, v) \in E$
- Goal: Find a hamiltonian cycle of G with minimum cost.

Solution space consists of at most $n!$ possible tours!

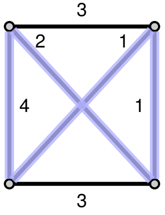
Actually the right number is $(n-1)!/2$



$2 + 4 + 1 + 1 = 8$

The Traveling Salesman Problem (TSP)

Given a set of cities along with the cost of travel between them, find the cheapest route visiting all cities and returning to your starting point.



$2 + 4 + 1 + 1 = 8$

Formal Definition

- Given: A complete undirected graph $G = (V, E)$ with nonnegative integer cost $c(u, v)$ for each edge $(u, v) \in E$
- Goal: Find a hamiltonian cycle of G with minimum cost.

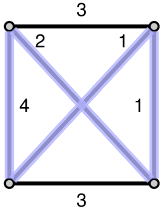
Solution space consists of at most $n!$ possible tours!

Actually the right number is $(n-1)!/2$

Special Instances

The Traveling Salesman Problem (TSP)

Given a set of cities along with the cost of travel between them, find the cheapest route visiting all cities and returning to your starting point.



$2 + 4 + 1 + 1 = 8$

Formal Definition

- Given: A complete undirected graph $G = (V, E)$ with nonnegative integer cost $c(u, v)$ for each edge $(u, v) \in E$
- Goal: Find a hamiltonian cycle of G with minimum cost.

Solution space consists of at most $n!$ possible tours!

Actually the right number is $(n-1)!/2$

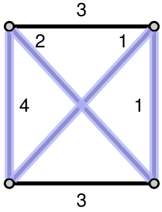
Special Instances

- Metric TSP: costs satisfy triangle inequality:

$$\forall u, v, w \in V: \qquad c(u, w) \leq c(u, v) + c(v, w).$$

The Traveling Salesman Problem (TSP)

Given a set of cities along with the cost of travel between them, find the cheapest route visiting all cities and returning to your starting point.



$2 + 4 + 1 + 1 = 8$

Formal Definition

- Given: A complete undirected graph $G = (V, E)$ with nonnegative integer cost $c(u, v)$ for each edge $(u, v) \in E$
- Goal: Find a hamiltonian cycle of G with minimum cost.

Solution space consists of at most $n!$ possible tours!

Actually the right number is $(n-1)!/2$

Special Instances

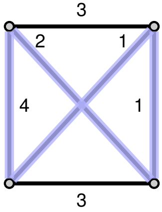
- Metric TSP: costs satisfy triangle inequality:

Even this version is NP hard (Ex. 35.2-2)

$$\forall u, v, w \in V: \qquad c(u, w) \leq c(u, v) + c(v, w).$$

The Traveling Salesman Problem (TSP)

Given a set of cities along with the cost of travel between them, find the cheapest route visiting all cities and returning to your starting point.



$2 + 4 + 1 + 1 = 8$

Formal Definition

- Given: A complete undirected graph $G = (V, E)$ with nonnegative integer cost $c(u, v)$ for each edge $(u, v) \in E$
- Goal: Find a hamiltonian cycle of G with minimum cost.

Solution space consists of at most $n!$ possible tours!

Actually the right number is $(n-1)!/2$

Special Instances

- Metric TSP: costs satisfy triangle inequality:

Even this version is NP hard (Ex. 35.2-2)

$$\forall u, v, w \in V: \qquad c(u, w) \le c(u, v) + c(v, w).$$

- Euclidean TSP: cities are points in the Euclidean space, costs are equal to their (rounded) Euclidean distance

Outline

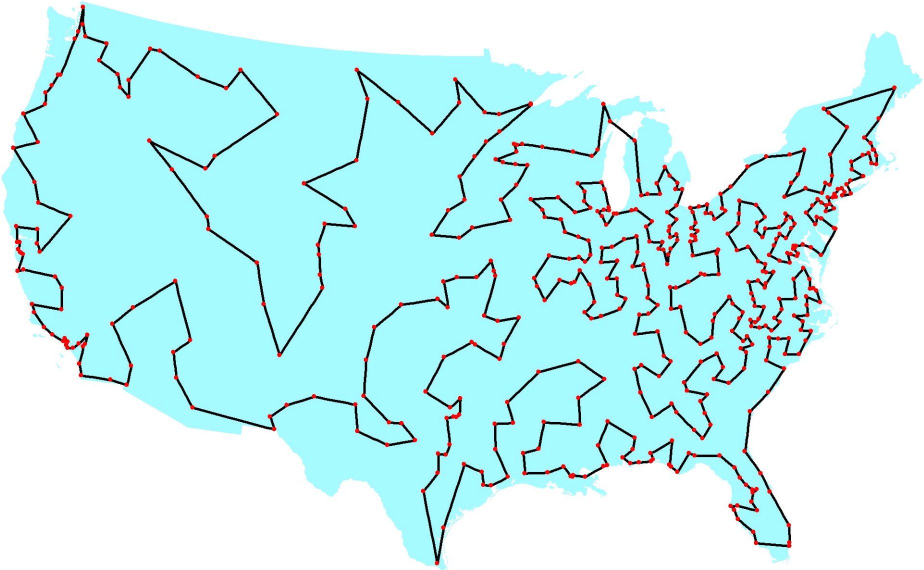
Introduction

Examples of TSP Instances

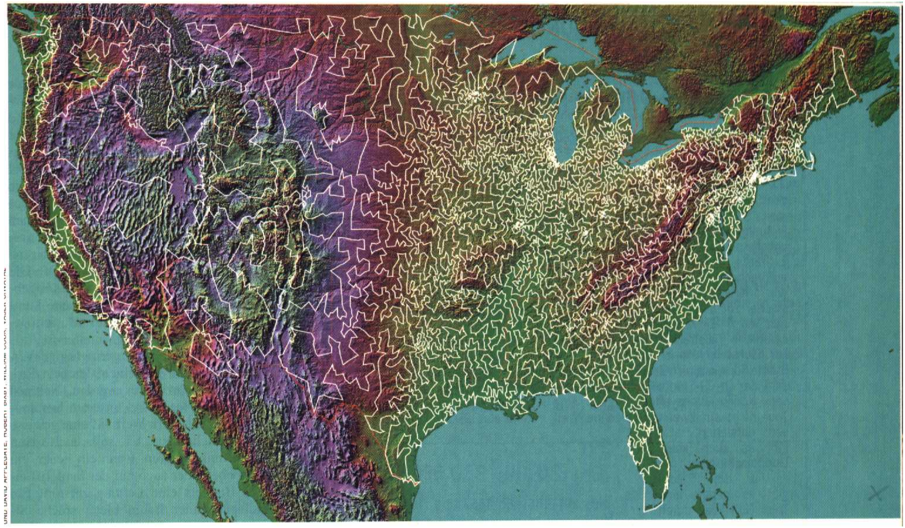
Demonstration

33 city contest (1964)

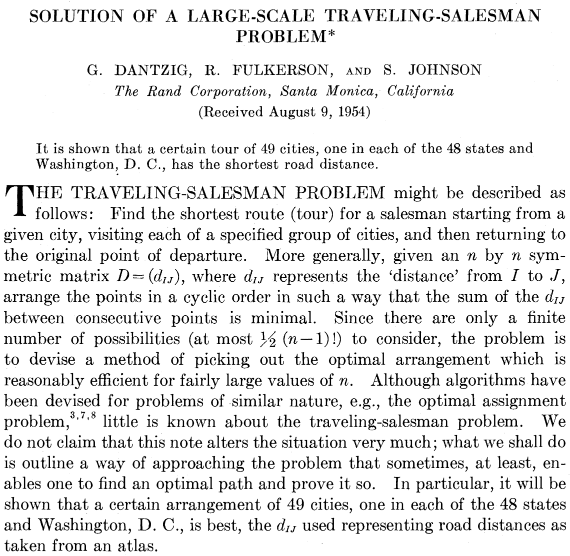
532 cities (1987 [Padberg, Rinaldi])



13,509 cities (1999 [Applegate, Bixby, Chavatal, Cook])



The Original Article (1954)


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.

The 42 (49) Cities

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.

Combinatorial Explosion

WolframAlpha computational intelligence.

(42-1)!/2

NATURAL LANGUAGE | MATH INPUT | EXTENDED KEYBOARD | EXAMPLES | UPLOAD | RANDOM

Input

$\frac{1}{2}(42-1)!$

$n!$ is the factorial function

Result

16 726 263 306 581 903 554 085 031 026 720 375 832 576 000 000 000

Scientific notation

$1.6726263306581903554085031026720375832576 \times 10^{49}$

Number name (Full name)

16 quindecillion ...

Number length

50 decimal digits

Alternative representations (More)

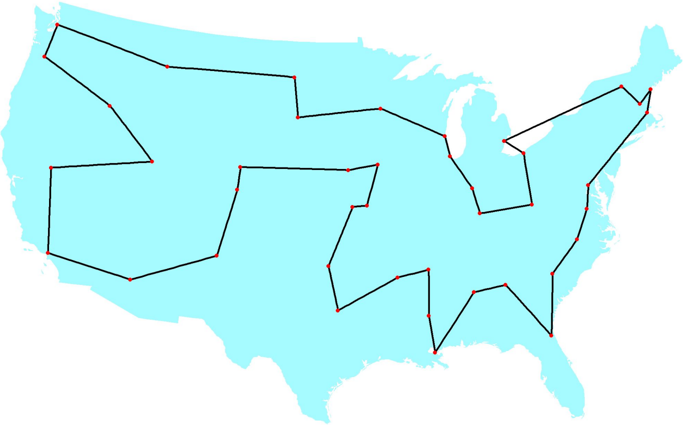
$$\frac{1}{2}(42-1)! = \frac{\Gamma(42)}{2}$$

$$\frac{1}{2}(42-1)! = \frac{\Gamma(42, 0)}{2}$$

$$\frac{1}{2}(42-1)! = \frac{(1)_{41}}{2}$$

Solution of this TSP problem

Dantzig, Fulkerson and Johnson found an optimal tour through 42 cities.



`http://www.math.uwaterloo.ca/tsp/history/img/dantzig_big.html`

Road Distances

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.

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
	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**

Road Distances

Hence this is an instance of the Metric TSP, but not Euclidean TSP.

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

Modelling TSP as a Linear Program Relaxation

Idea: Indicator variable $x(i, j)$, $i > j$, which is one if the tour includes edge $\{i, j\}$ (in either direction)

Modelling TSP as a Linear Program Relaxation

Idea: Indicator variable $x(i,j)$, $i > j$, which is one if the tour includes edge $\{i,j\}$ (in either direction)

minimize $\sum_{i=1}^{42} \sum_{j=1}^{i-1} c(i,j)x(i,j)$

subject to

$$\sum_{j<i} x(i,j) + \sum_{j>i} x(j,i) = 2 \qquad \text{for each } 1 \leq i \leq 42$$

$$0 \leq x(i,j) \leq 1 \qquad \text{for each } 1 \leq j < i \leq 42$$

Modelling TSP as a Linear Program Relaxation

Idea: Indicator variable $x(i,j)$, $i > j$, which is one if the tour includes edge $\{i,j\}$ (in either direction)

minimize $\quad \sum_{i=1}^{42} \sum_{j=1}^{i-1} c(i,j)x(i,j)$

subject to

$$\sum_{j<i} x(i,j) + \sum_{j>i} x(j,i) = 2 \qquad \text{for each } 1 \leq i \leq 42$$

$$0 \leq x(i,j) \leq 1 \qquad \text{for each } 1 \leq j < i \leq 42$$

Constraints $x(i,j) \in \{0,1\}$ are not allowed in a LP!

Modelling TSP as a Linear Program Relaxation

Idea: Indicator variable $x(i,j)$, $i > j$, which is one if the tour includes edge $\{i,j\}$ (in either direction)

minimize $\sum_{i=1}^{42} \sum_{j=1}^{i-1} c(i,j)x(i,j)$

subject to

$$\sum_{j<i} x(i,j) + \sum_{j>i} x(j,i) = 2 \qquad \text{for each } 1 \leq i \leq 42$$

$$0 \leq x(i,j) \leq 1 \qquad \text{for each } 1 \leq j < i \leq 42$$

Constraints $x(i,j) \in \{0,1\}$ are not allowed in a LP!

Branch & Bound to solve an Integer Program:

Modelling TSP as a Linear Program Relaxation

Idea: Indicator variable $x(i,j)$, $i > j$, which is one if the tour includes edge $\{i,j\}$ (in either direction)

minimize $\sum_{i=1}^{42} \sum_{j=1}^{i-1} c(i,j)x(i,j)$

subject to

$$\sum_{j<i} x(i,j) + \sum_{j>i} x(j,i) = 2 \qquad \text{for each } 1 \leq i \leq 42$$

$$0 \leq x(i,j) \leq 1 \qquad \text{for each } 1 \leq j < i \leq 42$$

Constraints $x(i,j) \in \{0,1\}$ are not allowed in a LP!

Branch & Bound to solve an Integer Program:

- As long as solution of LP has fractional $x(i,j) \in (0,1)$:

Modelling TSP as a Linear Program Relaxation

Idea: Indicator variable $x(i,j)$, $i > j$, which is one if the tour includes edge $\{i,j\}$ (in either direction)

$$\begin{array}{lll} \text{minimize} & \sum_{i=1}^{42} \sum_{j=1}^{i-1} c(i,j) x(i,j) & \\ \text{subject to} & & \\ & \sum_{j<i} x(i,j) + \sum_{j>i} x(j,i) = 2 & \text{for each } 1 \leq i \leq 42 \\ & 0 \leq x(i,j) \leq 1 & \text{for each } 1 \leq j < i \leq 42 \end{array}$$

Constraints $x(i,j) \in \{0,1\}$ are not allowed in a LP!

Branch & Bound to solve an Integer Program:

- As long as solution of LP has fractional $x(i,j) \in (0,1)$:
  - Add $x(i,j) = 0$ to the LP, solve it and recurse
  - Add $x(i,j) = 1$ to the LP, solve it and recurse
  - Return best of these two solutions

Modelling TSP as a Linear Program Relaxation

Idea: Indicator variable $x(i,j)$, $i > j$, which is one if the tour includes edge $\{i,j\}$ (in either direction)

$$\text{minimize} \qquad \sum_{i=1}^{42} \sum_{j=1}^{i-1} c(i,j)x(i,j)$$

subject to

$$\sum_{j<i} x(i,j) + \sum_{j>i} x(j,i) = 2 \qquad \text{for each } 1 \leq i \leq 42$$

$$0 \leq x(i,j) \leq 1 \qquad \text{for each } 1 \leq j < i \leq 42$$

Constraints $x(i,j) \in \{0,1\}$ are not allowed in a LP!

Branch & Bound to solve an Integer Program:

- As long as solution of LP has fractional $x(i,j) \in (0,1)$:
  - Add $x(i,j) = 0$ to the LP, solve it and recurse
  - Add $x(i,j) = 1$ to the LP, solve it and recurse
  - Return best of these two solutions
- If solution of LP integral, return objective value

Modelling TSP as a Linear Program Relaxation

Idea: Indicator variable $x(i,j)$, $i > j$, which is one if the tour includes edge $\{i,j\}$ (in either direction)

minimize $\qquad \sum_{i=1}^{42} \sum_{j=1}^{i-1} c(i,j)x(i,j)$

subject to

$$\sum_{j<i} x(i,j) + \sum_{j>i} x(j,i) = 2 \qquad \text{for each } 1 \leq i \leq 42$$

$$0 \leq x(i,j) \leq 1 \qquad \text{for each } 1 \leq j < i \leq 42$$

Constraints $x(i,j) \in \{0,1\}$ are not allowed in a LP!

Branch & Bound to solve an Integer Program:

- As long as solution of LP has fractional $x(i,j) \in (0,1)$:
  - Add $x(i,j) = 0$ to the LP, solve it and recurse
  - Add $x(i,j) = 1$ to the LP, solve it and recurse
  - Return best of these two solutions
- If solution of LP integral, return objective value

Bound-Step: If the best known integral solution so far is better than the solution of a LP, no need to explore branch further!

Outline

Introduction

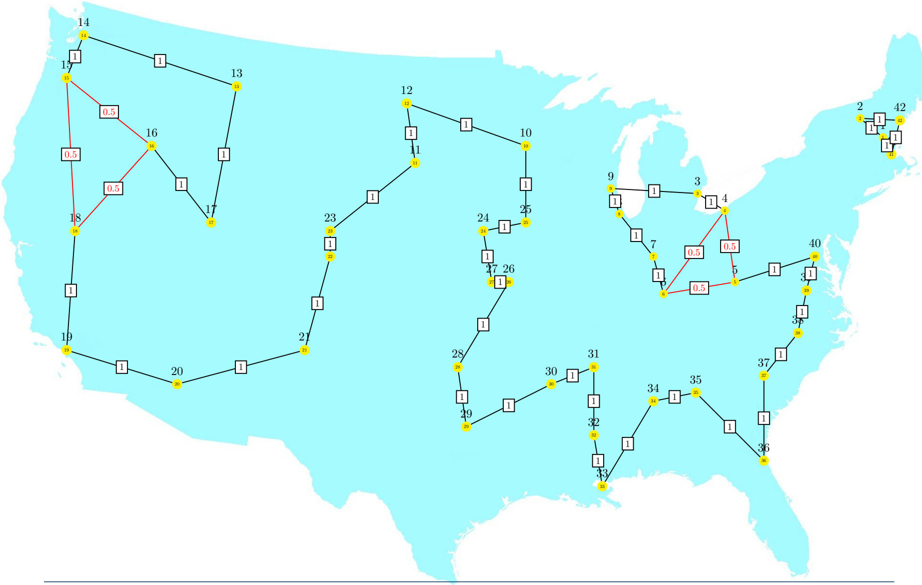
Examples of TSP Instances

Demonstration

In the following, there are a few different runs of the demo. In the our lecture, we choose a different branching variable in iteration 7 ($x_{12,13}$), and set it first to 0 and then to 1.

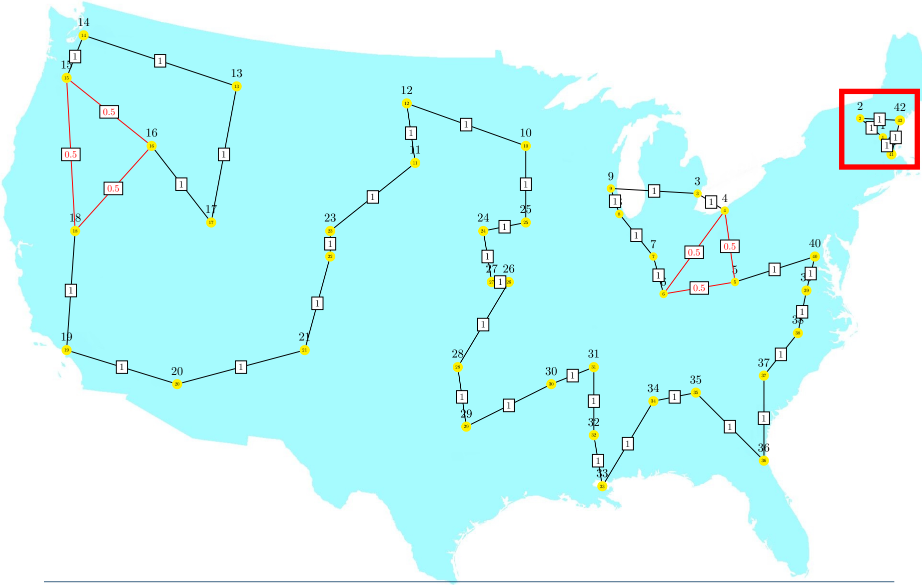
Iteration 1:

Objective value: -641.000000, 861 variables, 945 constraints, 1809 iterations



Iteration 1: Eliminate Subtour $1, 2, 41, 42$

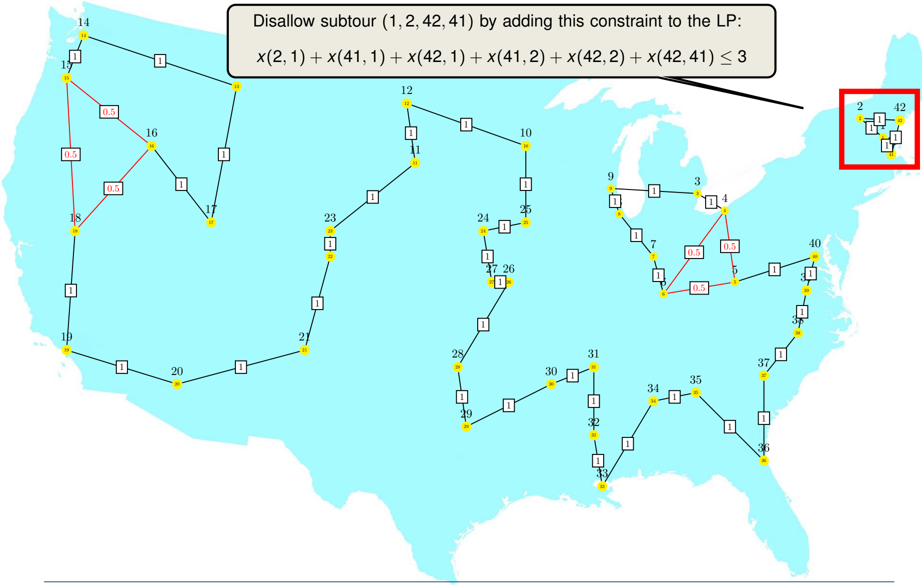
Objective value: -641.000000, 861 variables, 945 constraints, 1809 iterations



Objective value: -641.000000, 861 variables, 945 constraints, 1809 iterations

Disallow subtour $(1, 2, 42, 41)$ by adding this constraint to the LP:

$$x(2,1) + x(41,1) + x(42,1) + x(41,2) + x(42,2) + x(42,41) \leq 3$$



Iteration 1: Eliminate Subtour $1, 2, 41, 42$

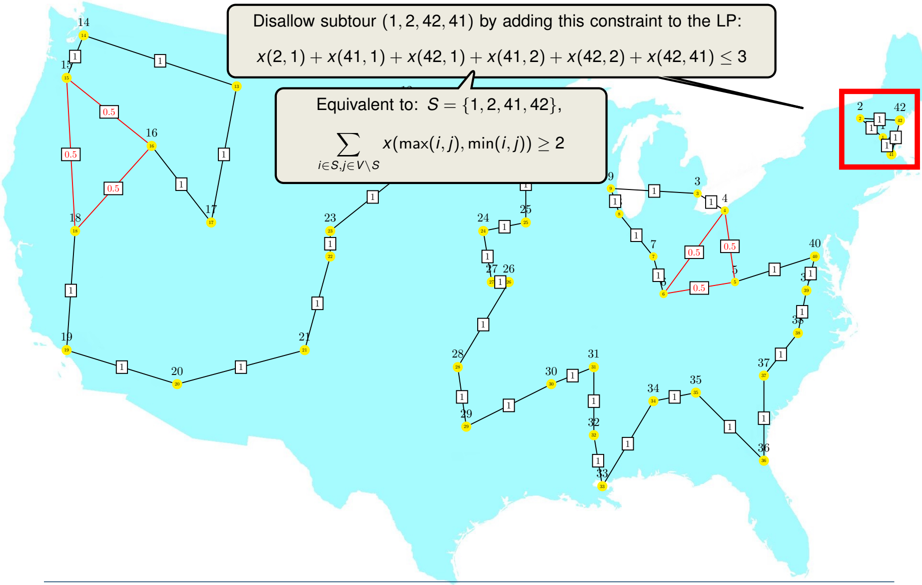
Objective value: -641.000000, 861 variables, 945 constraints, 1809 iterations

Disallow subtour $(1, 2, 42, 41)$ by adding this constraint to the LP:

$$x(2,1) + x(41,1) + x(42,1) + x(41,2) + x(42,2) + x(42,41) \leq 3$$

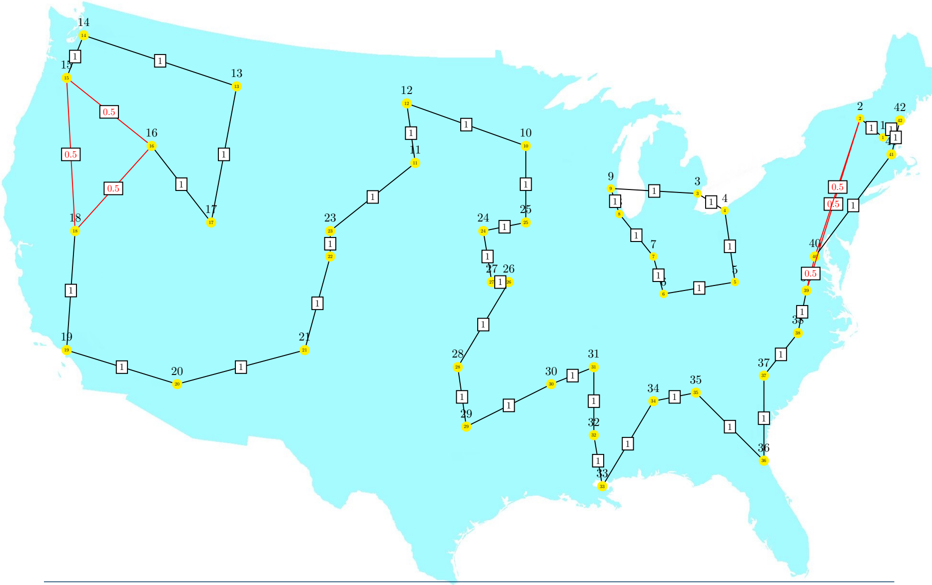
Equivalent to: $S = \{1, 2, 41, 42\}$,

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



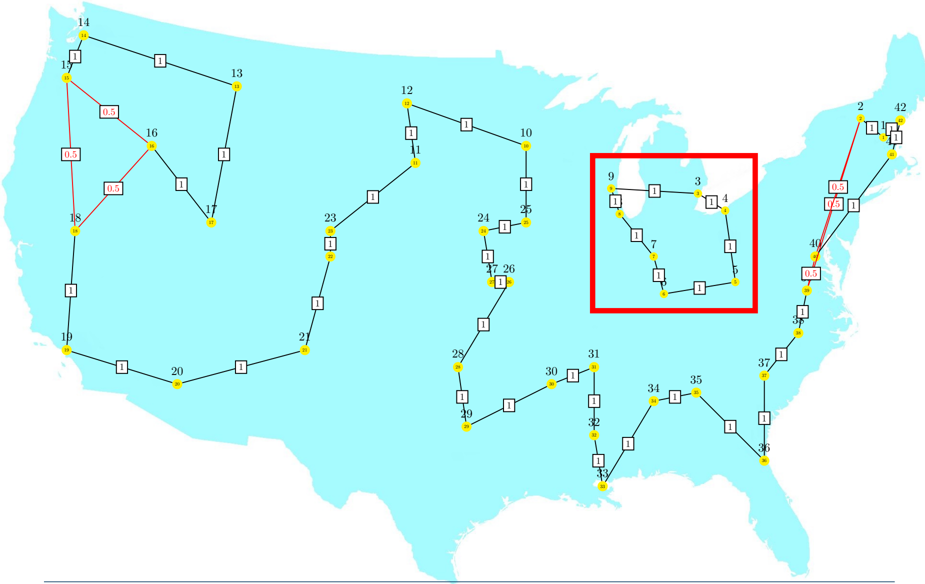
Iteration 2:

Objective value: -676.000000, 861 variables, 946 constraints, 1802 iterations



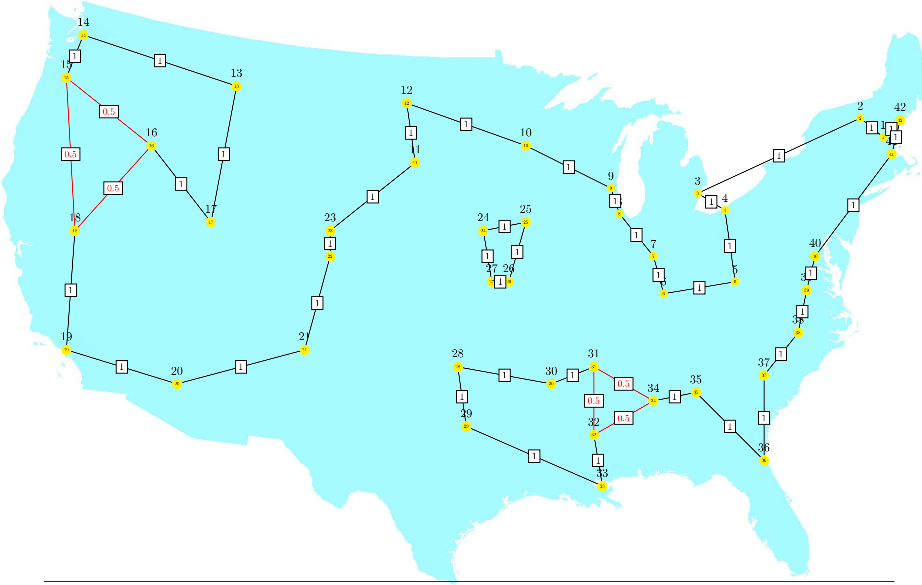
Iteration 2: Eliminate Subtour $3 - 9$

Objective value: -676.000000, 861 variables, 946 constraints, 1802 iterations



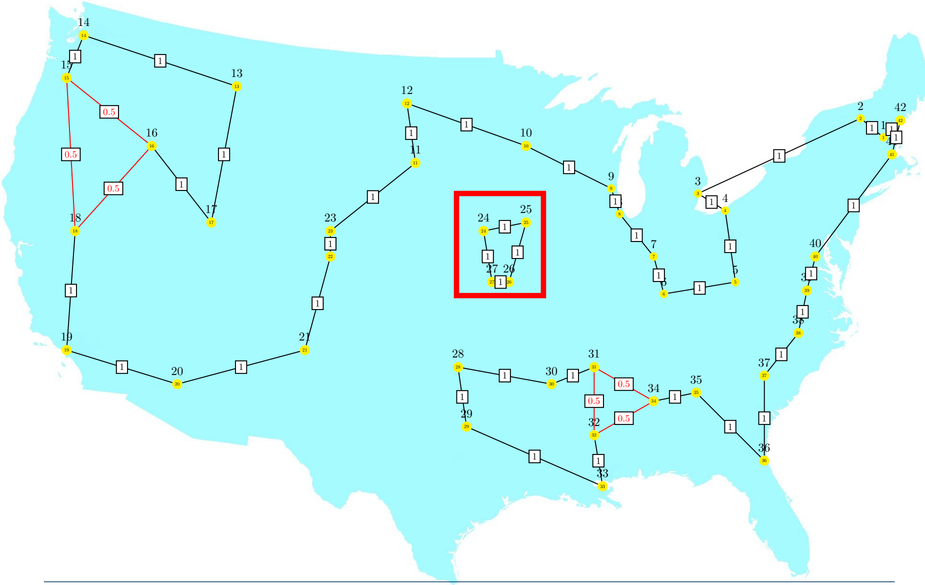
Iteration 3:

Objective value: -681.000000, 861 variables, 947 constraints, 1984 iterations



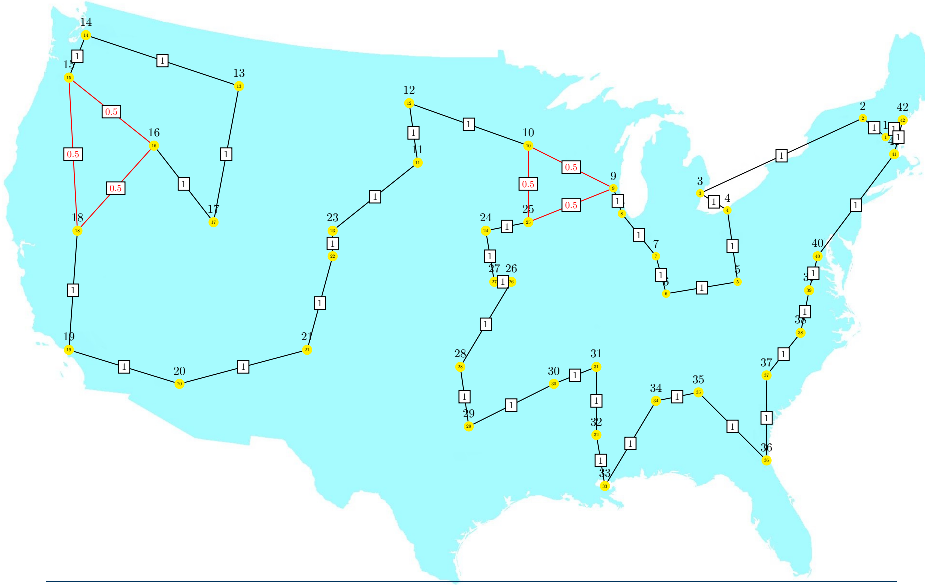
Iteration 3: Eliminate Subtour $24, 25, 26, 27$

Objective value: -681.000000, 861 variables, 947 constraints, 1984 iterations



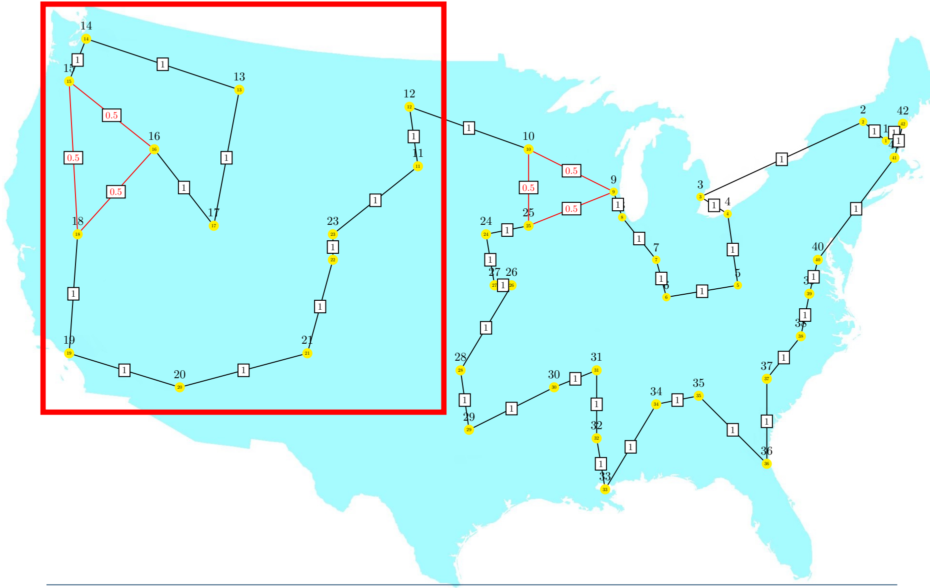
Iteration 4:

Objective value: -682.500000, 861 variables, 948 constraints, 1492 iterations



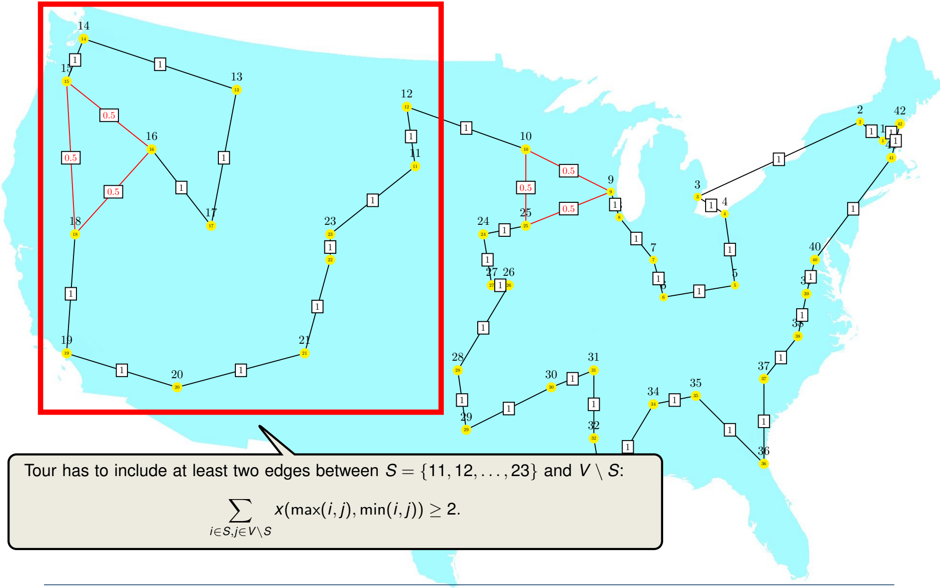
Iteration 4: Eliminate Cut $11 - 23$

Objective value: -682.500000, 861 variables, 948 constraints, 1492 iterations



Iteration 4: Eliminate Cut $11 - 23$

Objective value: -682.500000, 861 variables, 948 constraints, 1492 iterations

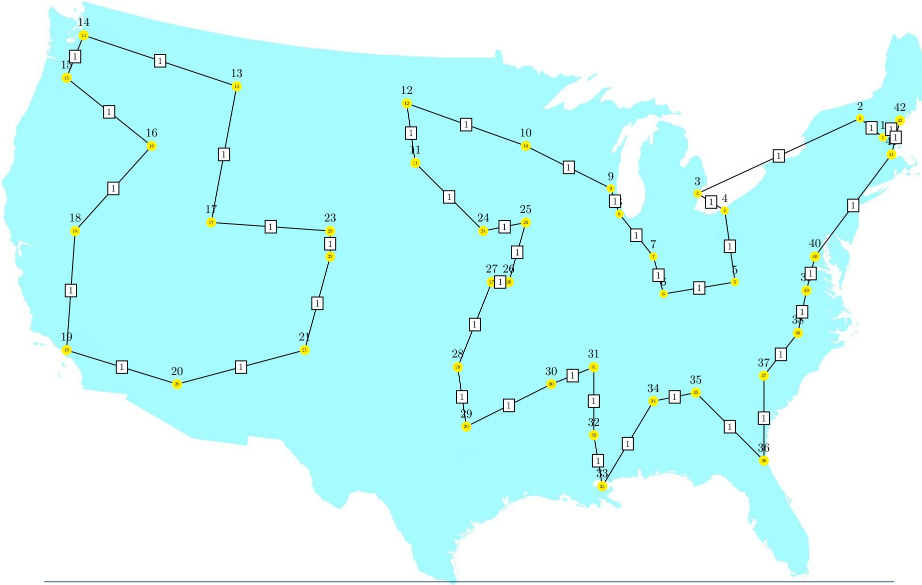


Tour has to include at least two edges between $S = \{11, 12, \ldots, 23\}$ and $V \setminus S$:

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

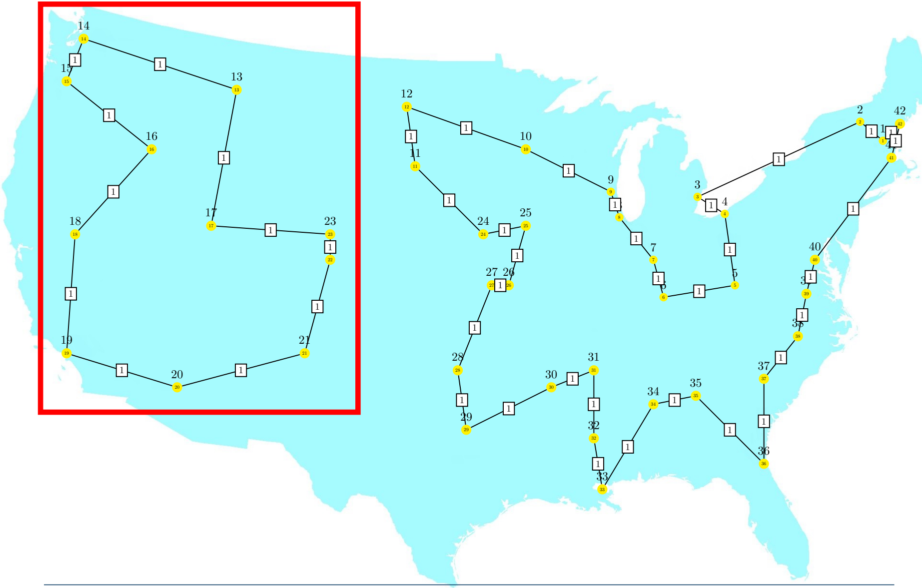
Iteration 5:

Objective value: -686.000000, 861 variables, 949 constraints, 2446 iterations



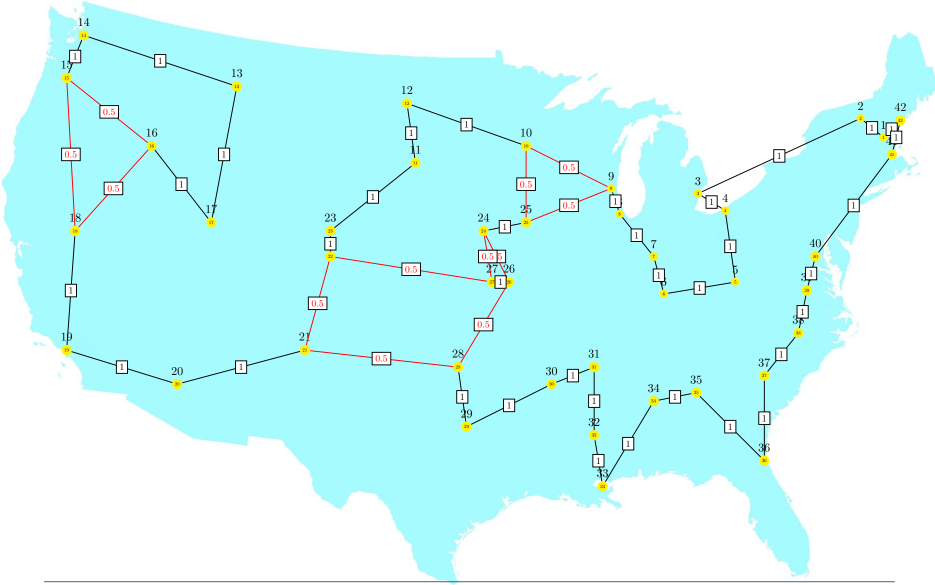
Iteration 5: Eliminate Subtour $13 - 23$

Objective value: -686.000000, 861 variables, 949 constraints, 2446 iterations



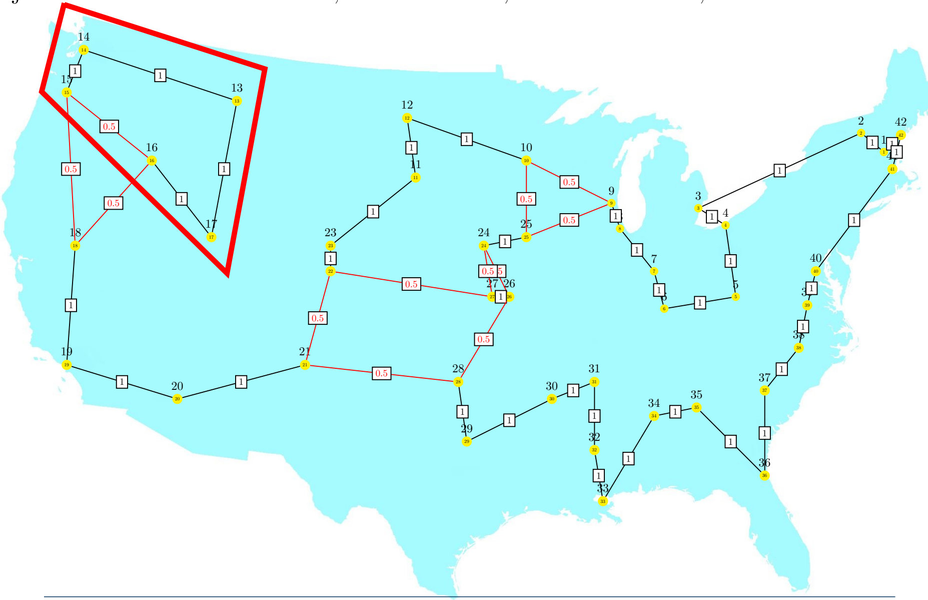
Iteration 6:

Objective value: -694.500000, 861 variables, 950 constraints, 1690 iterations



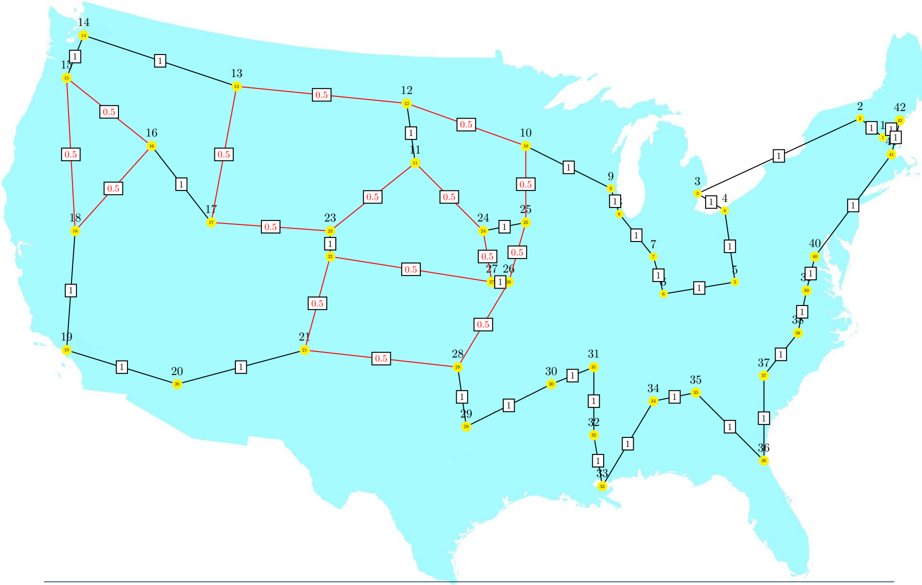
Iteration 6: Eliminate Cut $13 - 17$

Objective value: -694.500000, 861 variables, 950 constraints, 1690 iterations



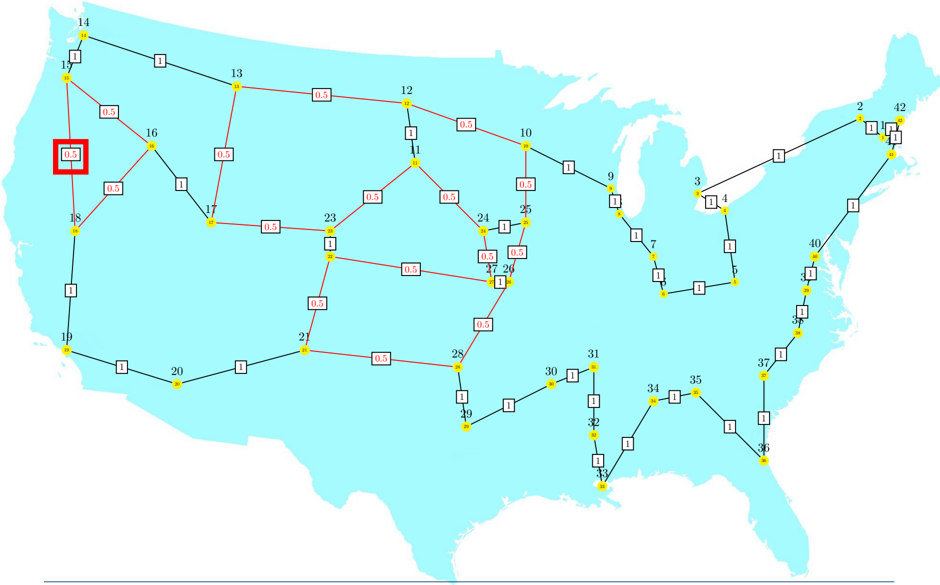
Iteration 7:

Objective value: -697.000000, 861 variables, 951 constraints, 2212 iterations



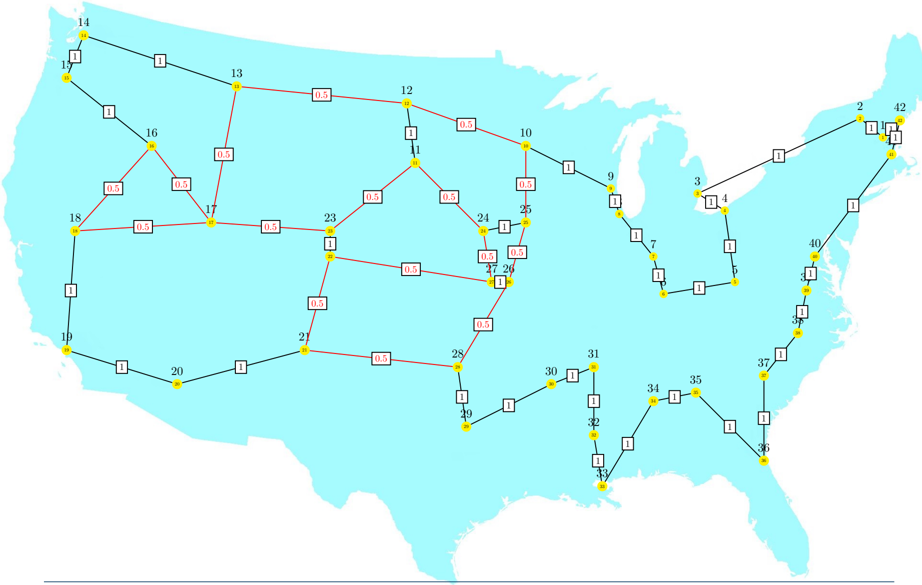
Iteration 7: Branch 1a $x_{18,15} = 0$

Objective value: -697.000000, 861 variables, 951 constraints, 2212 iterations



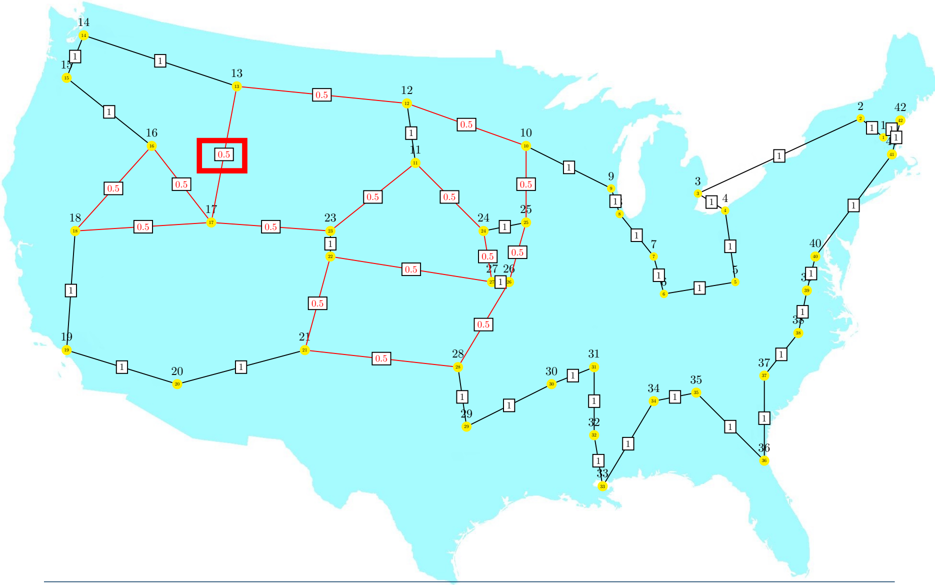
Iteration 8:

Objective value: -698.000000, 861 variables, 952 constraints, 1878 iterations



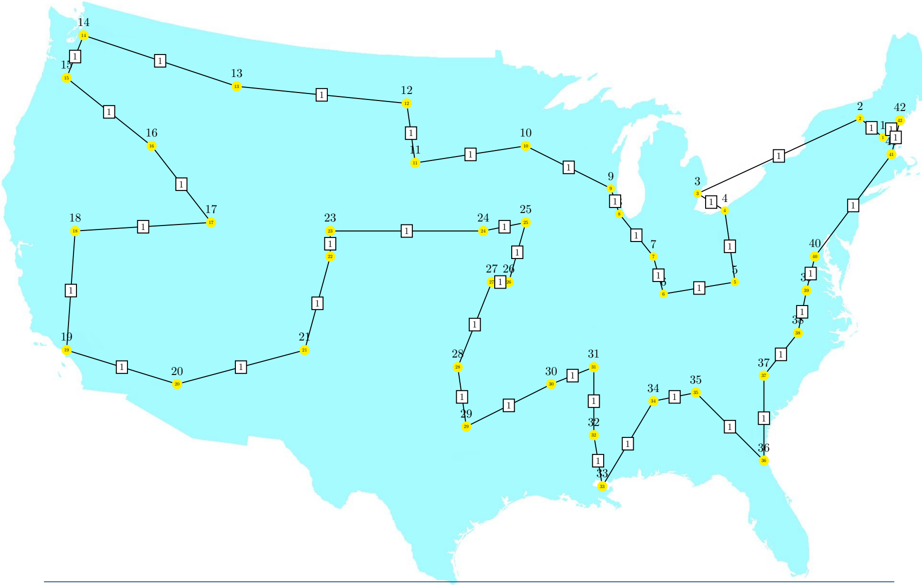
Iteration 8: Branch 2a $x_{17,13} = 0$

Objective value: -698.000000, 861 variables, 952 constraints, 1878 iterations



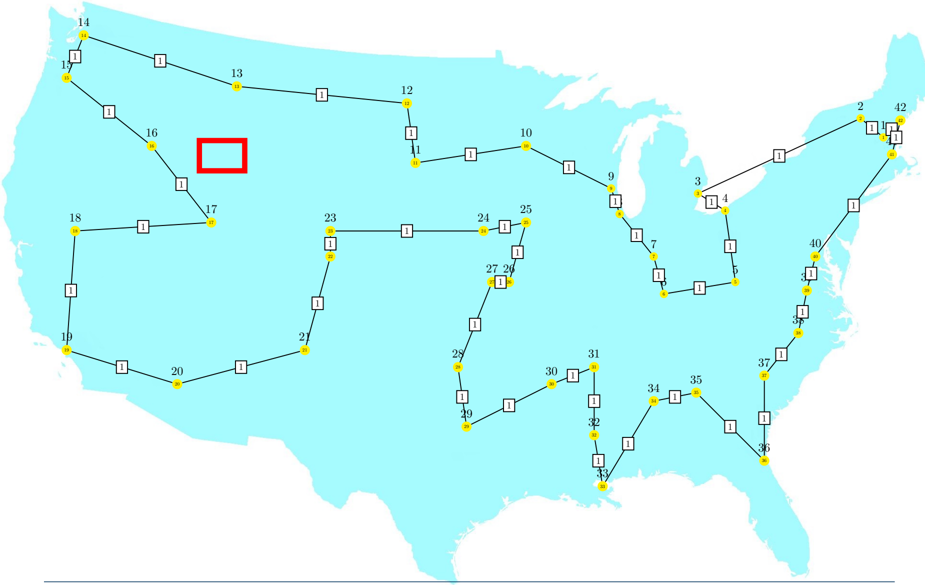
Iteration 9:

Objective value: −699.000000, 861 variables, 953 constraints, 2281 iterations



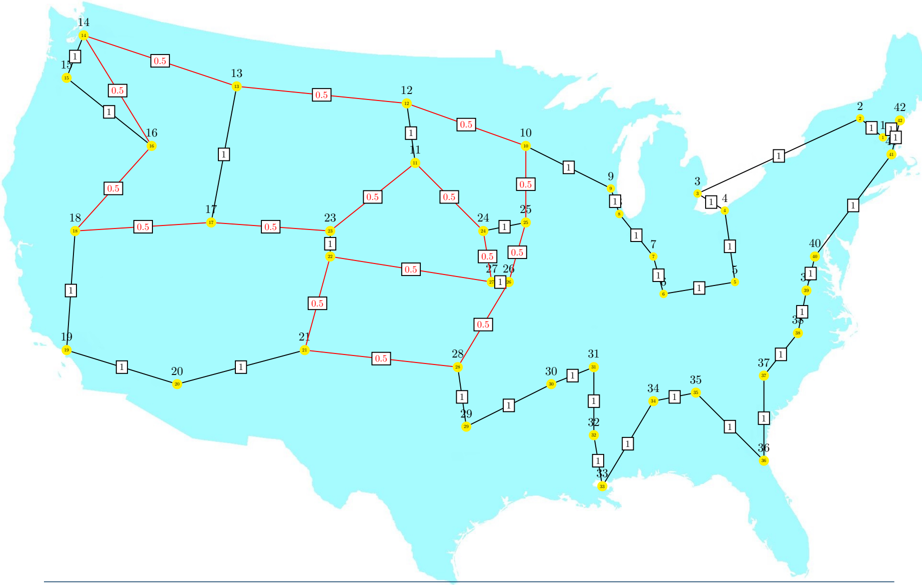
Iteration 9: Branch 2b $x_{17,13} = 1$

Objective value: -699.000000, 861 variables, 953 constraints, 2281 iterations



Iteration 10:

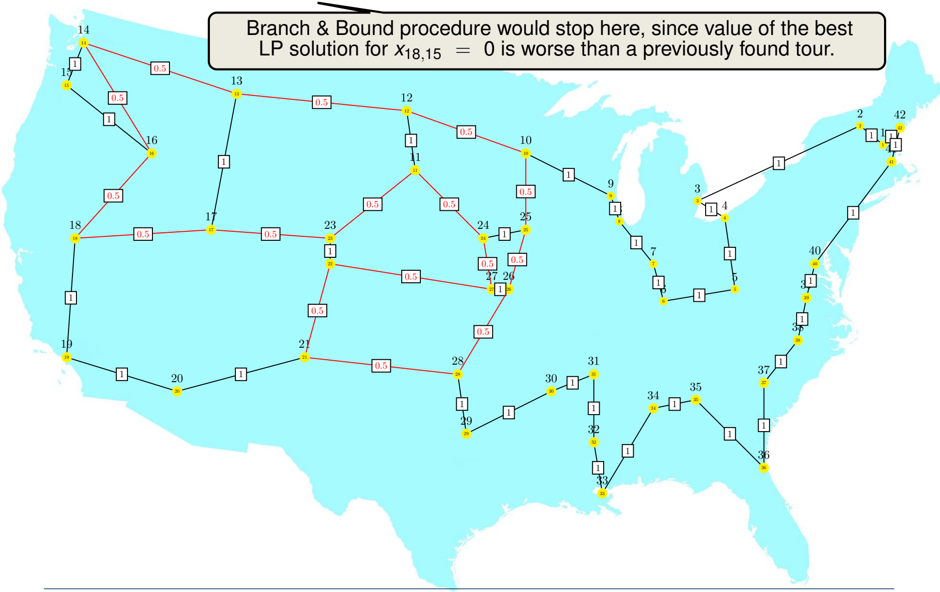
Objective value: -700.000000, 861 variables, 954 constraints, 2398 iterations



Iteration 10:

Objective value: -700.000000, 861 variables, 954 constraints, 2398 iterations

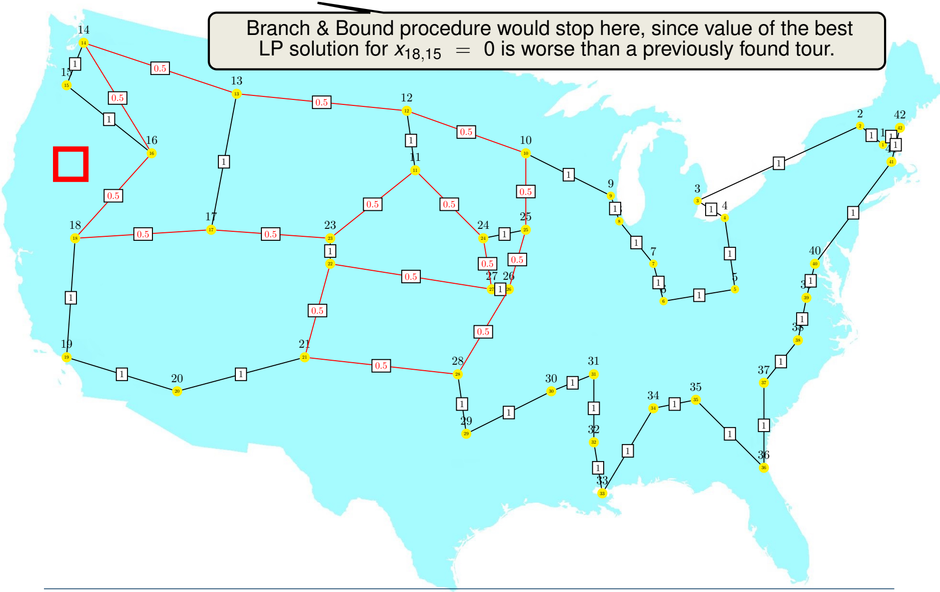
Branch & Bound procedure would stop here, since value of the best LP solution for $x_{18,15} = 0$ is worse than a previously found tour.



Iteration 10: Branch 1b $x_{18,15} = 1$

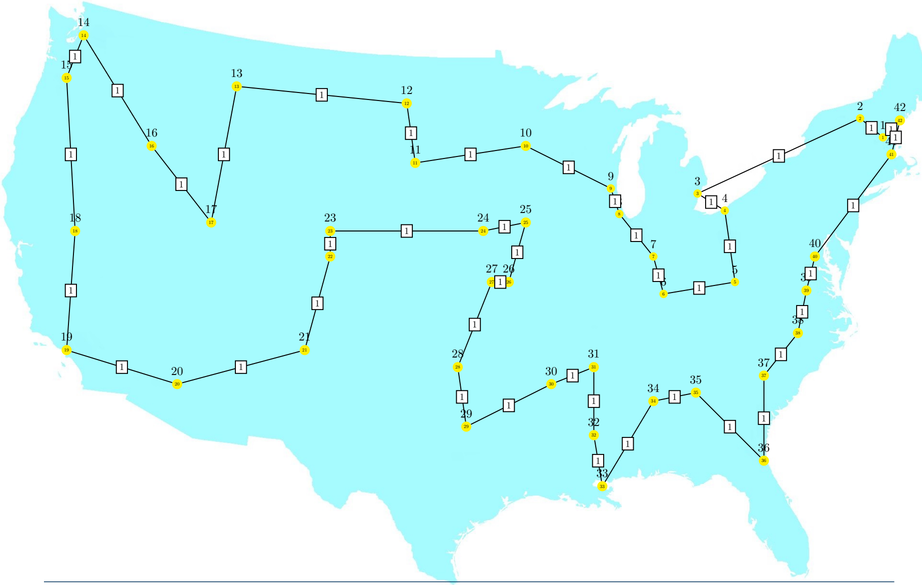
Objective value: -700.000000, 861 variables, 954 constraints, 2398 iterations

Branch & Bound procedure would stop here, since value of the best LP solution for $x_{18,15} = 0$ is worse than a previously found tour.



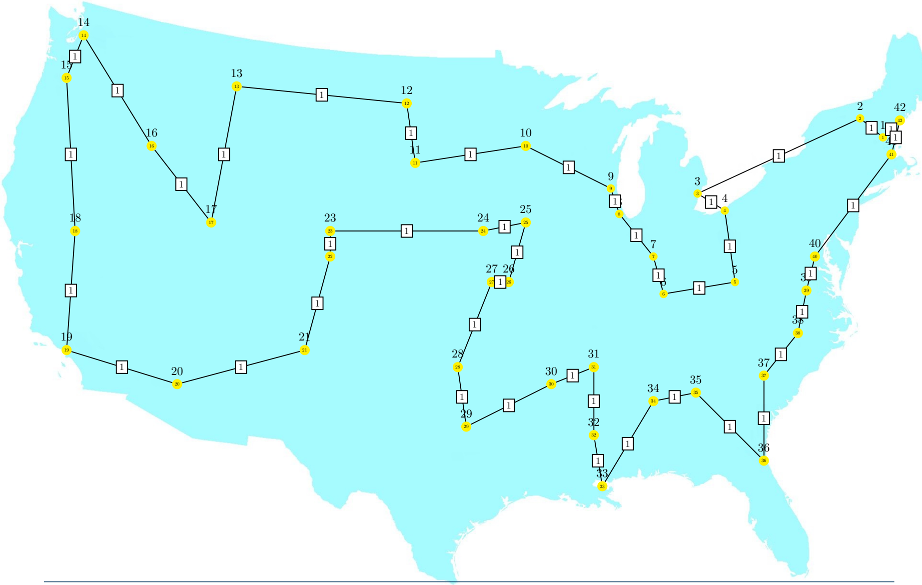
Iteration 11:

Objective value: -701.000000, 861 variables, 953 constraints, 2506 iterations



Iteration 11: Branch & Bound terminates

Objective value: -701.000000, 861 variables, 953 constraints, 2506 iterations



Branch & Bound Overview

1: LP solution 641

Branch & Bound Overview

1: LP solution 641

Eliminate Subtour $1, 2, 41, 42$

Branch & Bound Overview

1: LP solution 641

$\downarrow$ Eliminate Subtour $1, 2, 41, 42$

2: LP solution 676

Branch & Bound Overview

1: LP solution 641

Eliminate Subtour $1, 2, 41, 42$

2: LP solution 676

Eliminate Subtour $3 - 9$

Branch & Bound Overview

1: LP solution 641

↓ Eliminate Subtour $1, 2, 41, 42$

2: LP solution 676

↓ Eliminate Subtour $3 - 9$

3: LP solution 681

Branch & Bound Overview

1: LP solution 641

Eliminate Subtour $1, 2, 41, 42$

2: LP solution 676

Eliminate Subtour $3 - 9$

3: LP solution 681

Eliminate Subtour $24, 25, 26, 27$

Branch & Bound Overview

1: LP solution 641

↓ Eliminate Subtour $1, 2, 41, 42$

2: LP solution 676

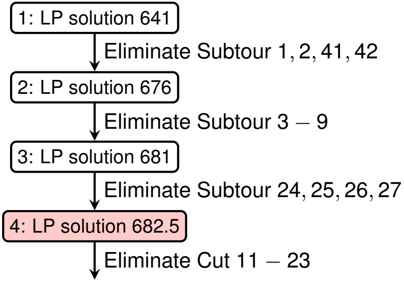
↓ Eliminate Subtour $3 - 9$

3: LP solution 681

↓ Eliminate Subtour $24, 25, 26, 27$

4: LP solution 682.5

Branch & Bound Overview



Branch & Bound Overview

$$
\begin{array}{c}
\boxed{\text{1: LP solution 641}} \\
\big\downarrow \ \text{Eliminate Subtour } 1, 2, 41, 42 \\
\boxed{\text{2: LP solution 676}} \\
\big\downarrow \ \text{Eliminate Subtour } 3 - 9 \\
\boxed{\text{3: LP solution 681}} \\
\big\downarrow \ \text{Eliminate Subtour } 24, 25, 26, 27 \\
\boxed{\text{4: LP solution 682.5}} \\
\big\downarrow \ \text{Eliminate Cut } 11 - 23 \\
\boxed{\text{5: LP solution 686}}
\end{array}
$$

Branch & Bound Overview

1: LP solution 641

$\downarrow$ Eliminate Subtour $1, 2, 41, 42$

2: LP solution 676

$\downarrow$ Eliminate Subtour $3 - 9$

3: LP solution 681

$\downarrow$ Eliminate Subtour $24, 25, 26, 27$

4: LP solution 682.5

$\downarrow$ Eliminate Cut $11 - 23$

5: LP solution 686

$\downarrow$ Eliminate Subtour $10, 11, 12$

Branch & Bound Overview

1: LP solution 641

↓ Eliminate Subtour $1, 2, 41, 42$

2: LP solution 676

↓ Eliminate Subtour $3 - 9$

3: LP solution 681

↓ Eliminate Subtour $24, 25, 26, 27$

4: LP solution 682.5

↓ Eliminate Cut $11 - 23$

5: LP solution 686

↓ Eliminate Subtour $10, 11, 12$

6: LP solution 694.5

Branch & Bound Overview

1: LP solution 641

↓ Eliminate Subtour $1, 2, 41, 42$

2: LP solution 676

↓ Eliminate Subtour $3 - 9$

3: LP solution 681

↓ Eliminate Subtour $24, 25, 26, 27$

4: LP solution 682.5

↓ Eliminate Cut $11 - 23$

5: LP solution 686

↓ Eliminate Subtour $10, 11, 12$

6: LP solution 694.5

↓ Eliminate Cut $13 - 17$

Branch & Bound Overview

1: LP solution 641

↓ Eliminate Subtour $1, 2, 41, 42$

2: LP solution 676

↓ Eliminate Subtour $3 - 9$

3: LP solution 681

↓ Eliminate Subtour $24, 25, 26, 27$

4: LP solution 682.5

↓ Eliminate Cut $11 - 23$

5: LP solution 686

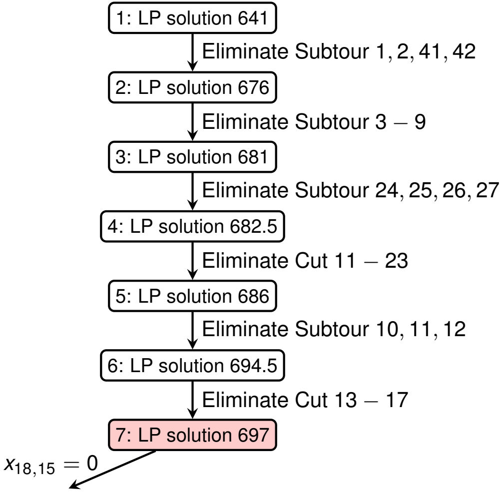
↓ Eliminate Subtour $10, 11, 12$

6: LP solution 694.5

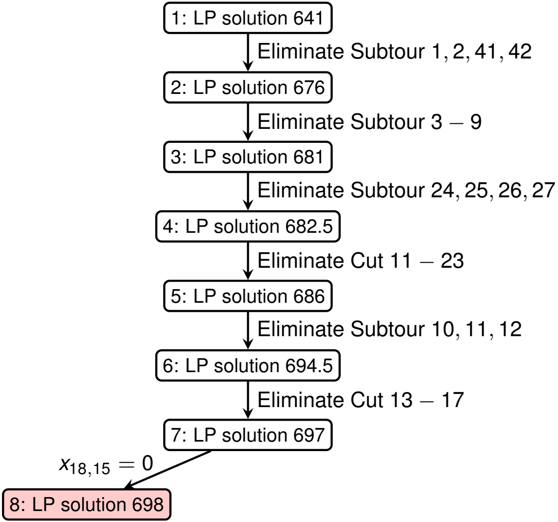
↓ Eliminate Cut $13 - 17$

7: LP solution 697

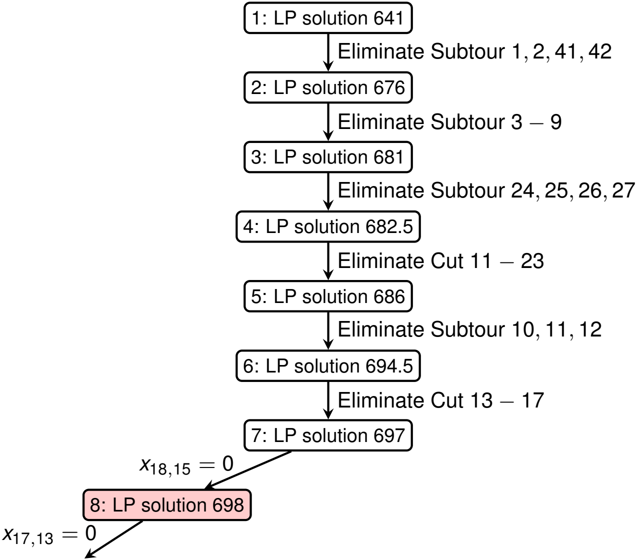
Branch & Bound Overview



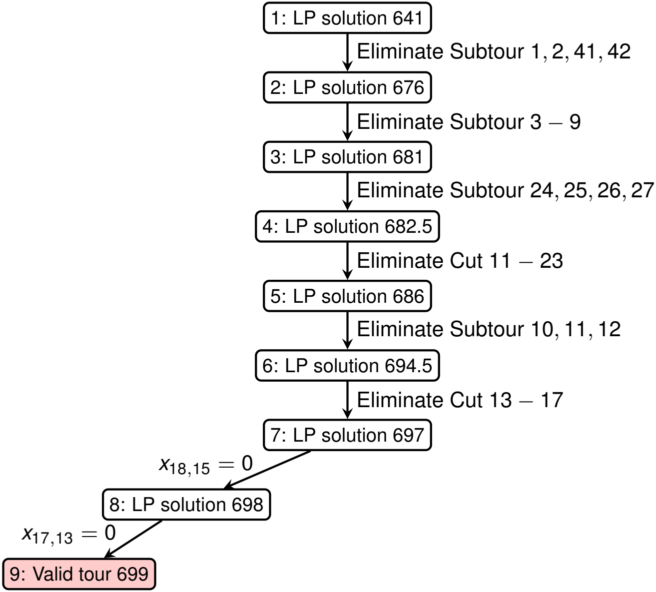
Branch & Bound Overview



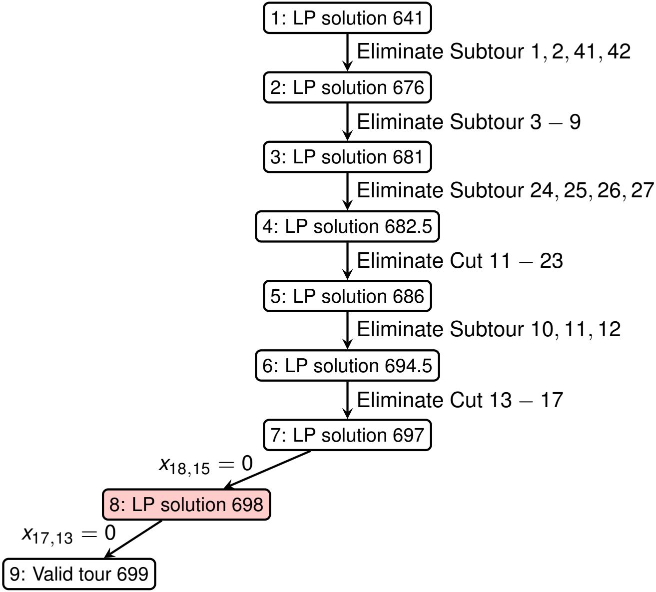
Branch & Bound Overview



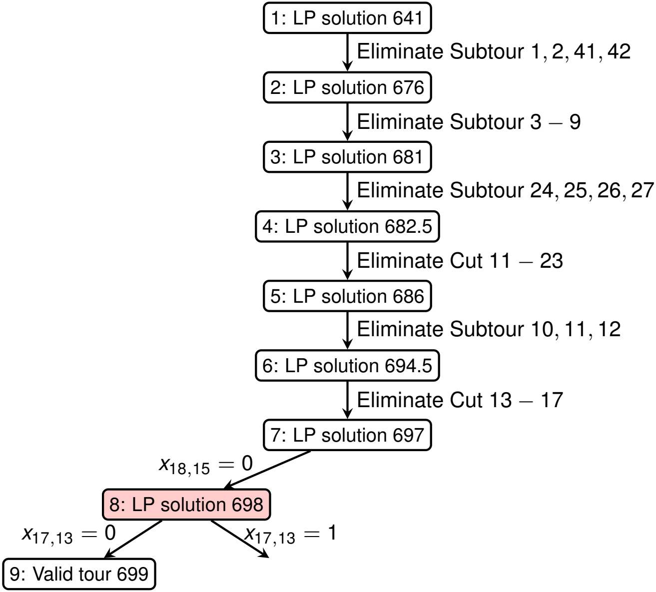
Branch & Bound Overview



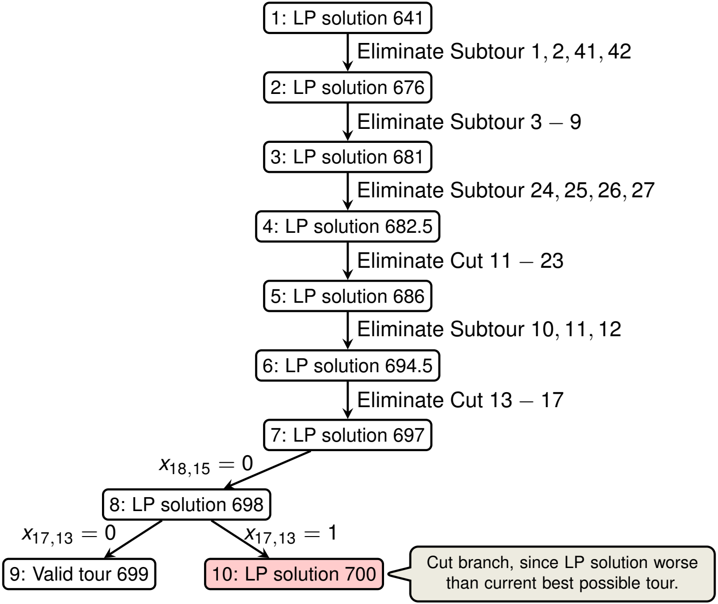
Branch & Bound Overview



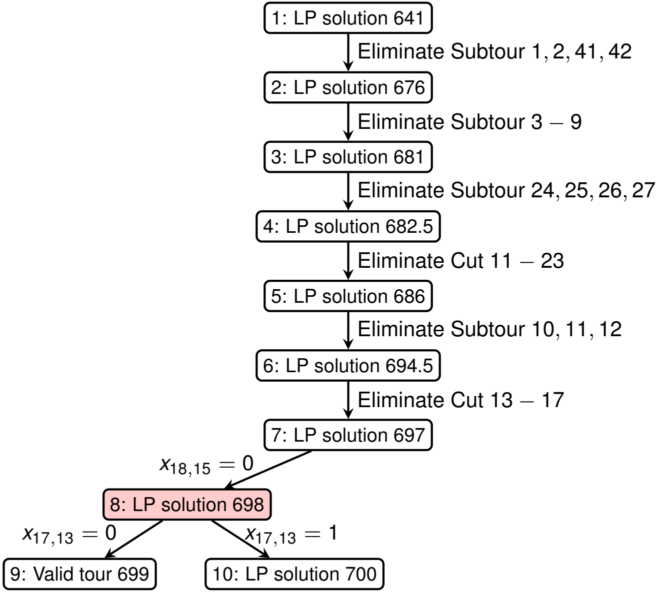
Branch & Bound Overview



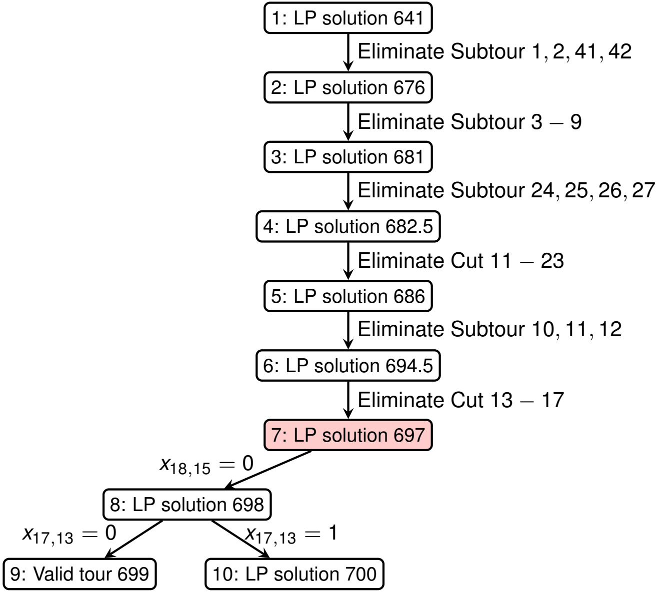
Branch & Bound Overview



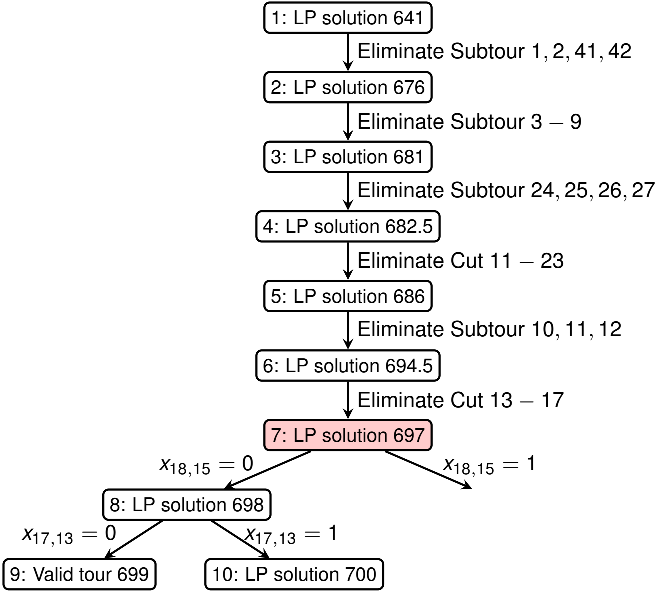
Branch & Bound Overview



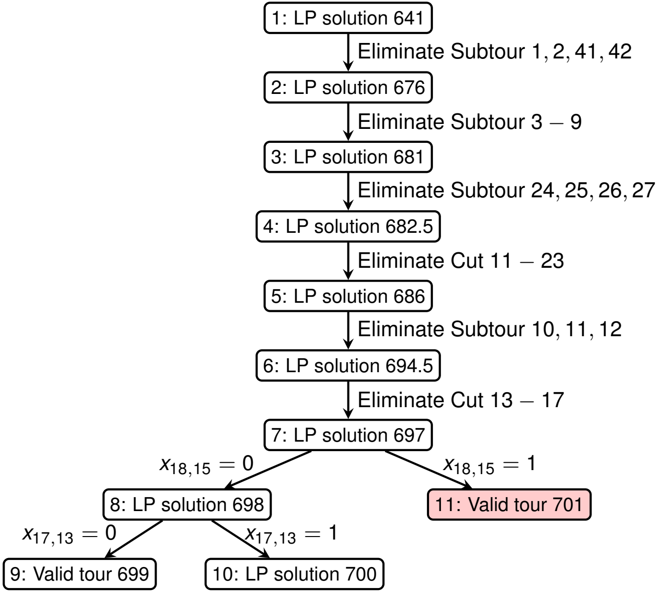
Branch & Bound Overview



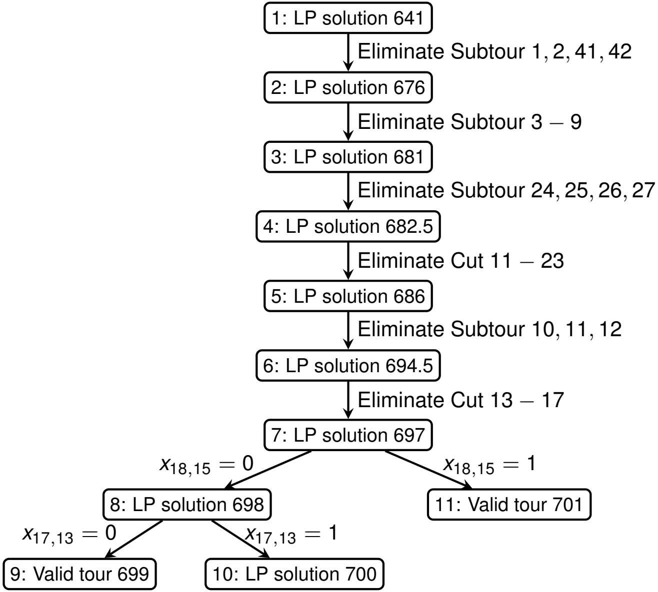
Branch & Bound Overview



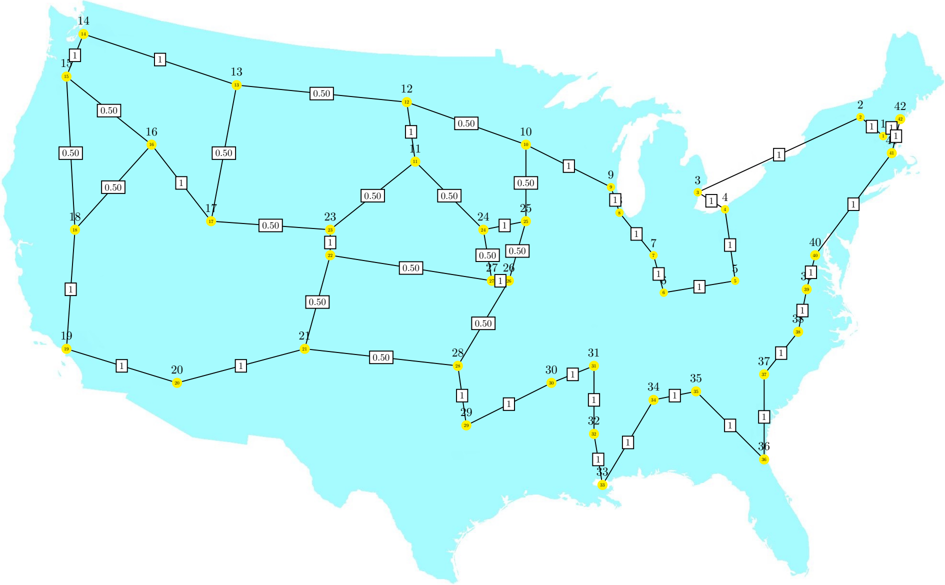
Branch & Bound Overview



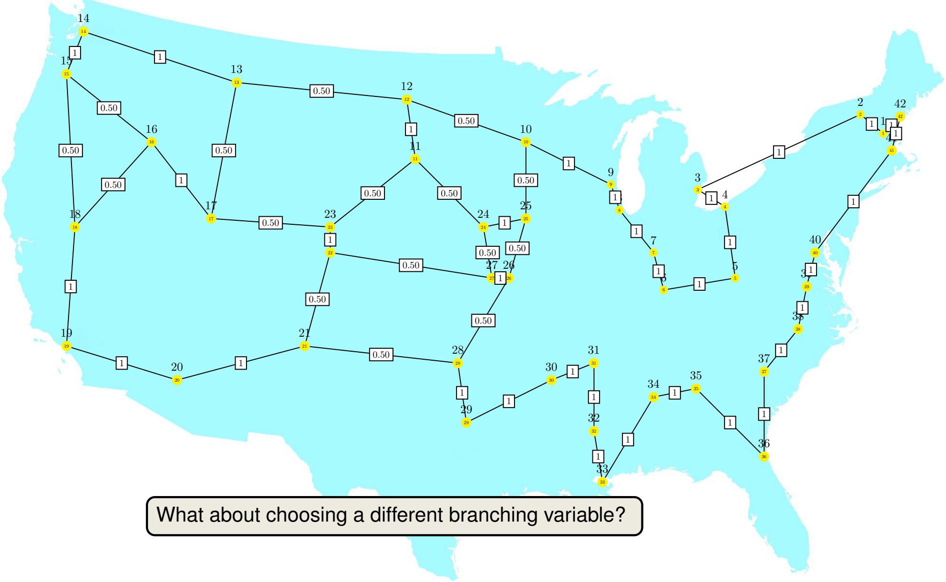
Branch & Bound Overview



Iteration 7: Objective 697



Iteration 7: Objective 697



What about choosing a different branching variable?

Solving Progress (Alternative Branch 1)

1: LP solution 641

↓ Eliminate Subtour $1, 2, 41, 42$

2: LP solution 676

↓ Eliminate Subtour $3 - 9$

3: LP solution 681

↓ Eliminate Subtour $24, 25, 26, 27$

4: LP solution 682.5

↓ Eliminate Cut $13 - 17$

5: LP solution 686

↓ Eliminate Subtour $10, 11, 12$

6: LP solution 686

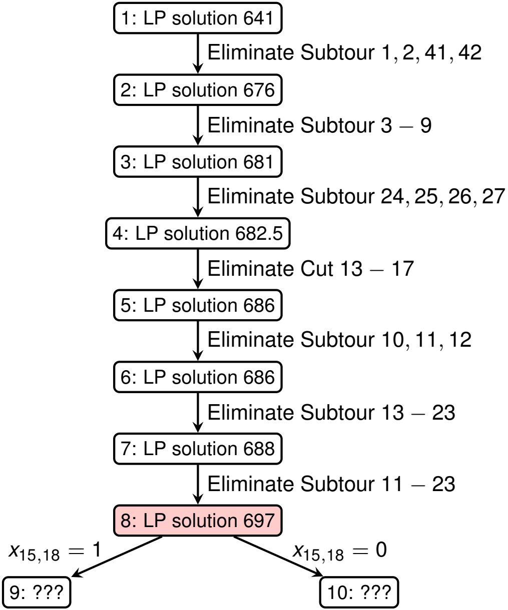
↓ Eliminate Subtour $13 - 23$

7: LP solution 688

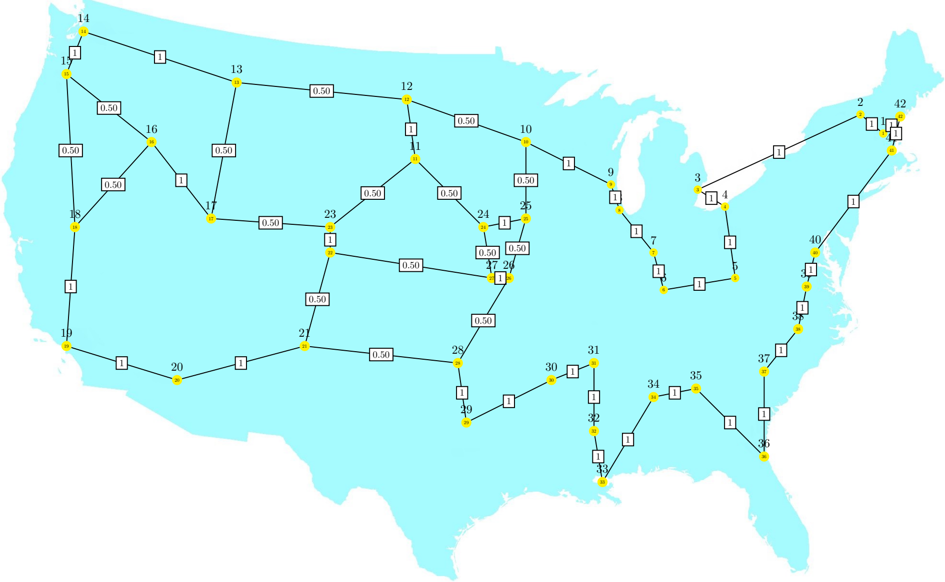
↓ Eliminate Subtour $11 - 23$

8: LP solution 697

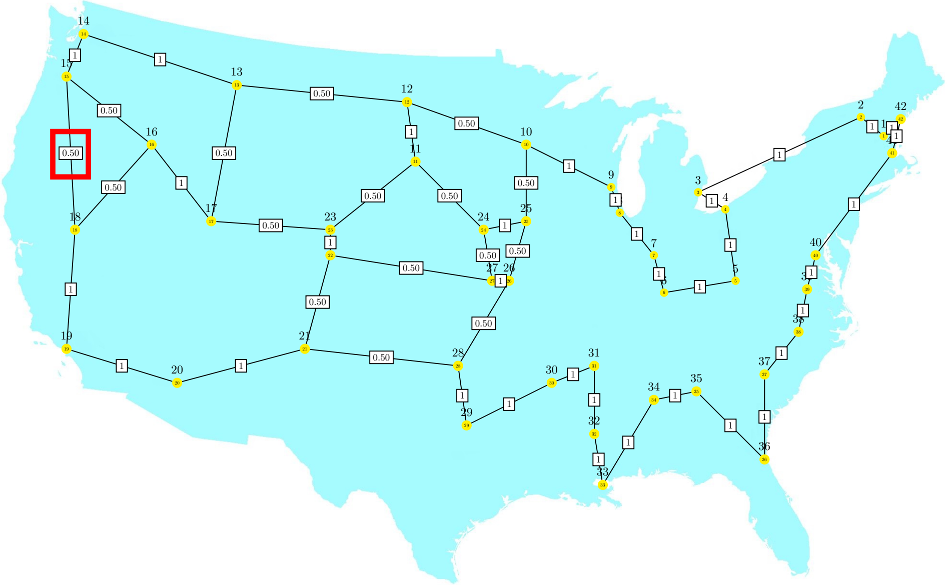
Solving Progress (Alternative Branch 1)



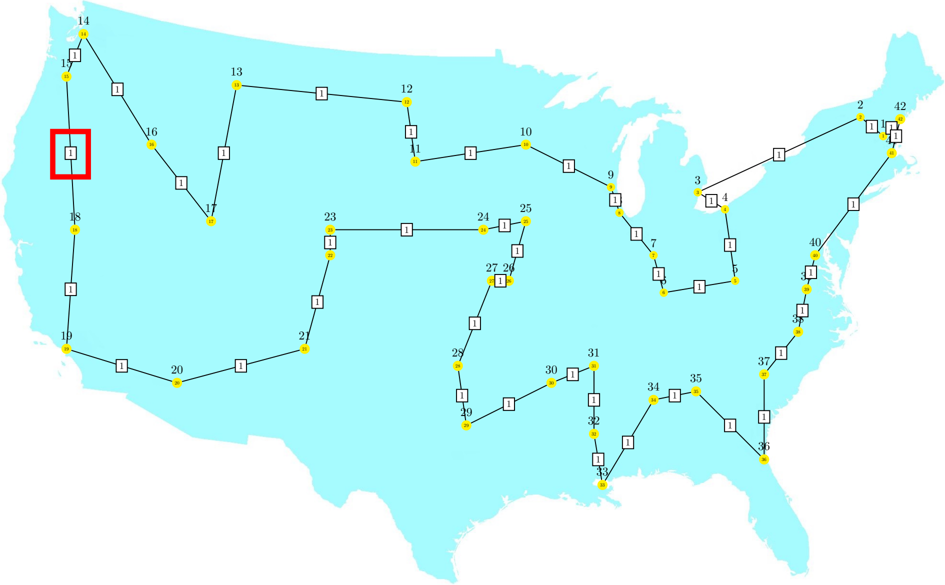
Alternative Branch 1: $x_{18,15}$, Objective 697



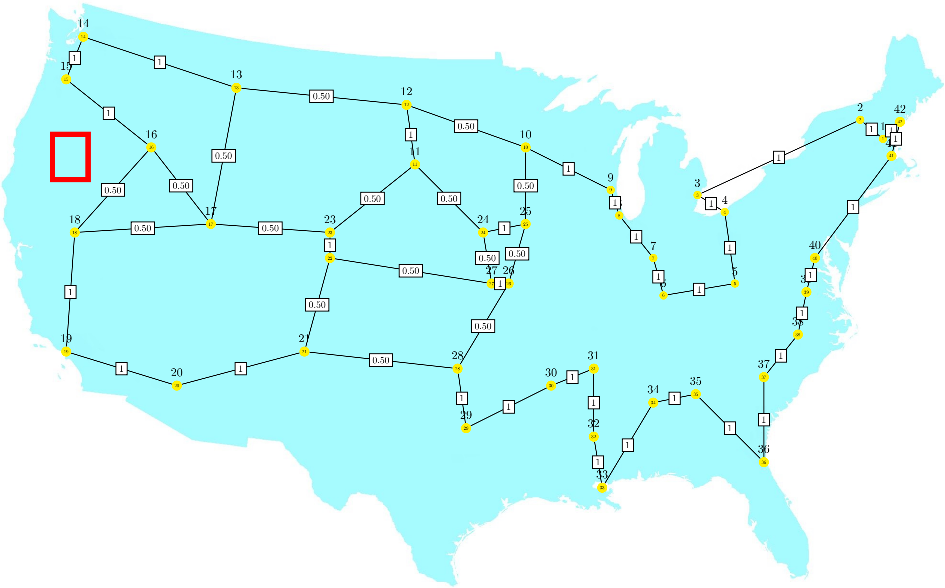
Alternative Branch 1: $x_{18,15}$, Objective 697



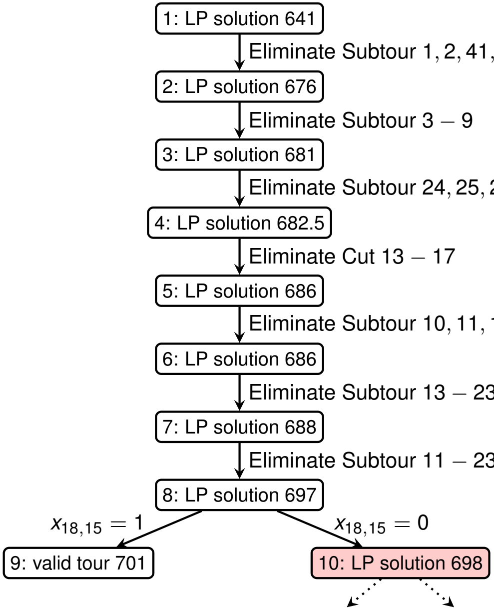
Alternative Branch 1a: $x_{18,15} = 1$, Objective 701 (Valid Tour)



Alternative Branch 1b: $x_{18,15} = 0$, Objective 698



Solving Progress (Alternative Branch 1)



Solving Progress (Alternative Branch 2)

1: LP solution 641

↓ Eliminate Subtour $1, 2, 41, 42$

2: LP solution 676

↓ Eliminate Subtour $3 - 9$

3: LP solution 681

↓ Eliminate Subtour $24, 25, 26, 27$

4: LP solution 682.5

↓ Eliminate Cut $13 - 17$

5: LP solution 686

↓ Eliminate Subtour $10, 11, 12$

6: LP solution 686

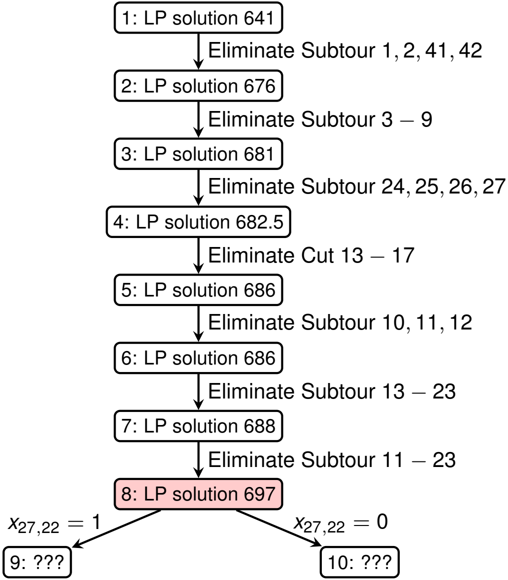
↓ Eliminate Subtour $13 - 23$

7: LP solution 688

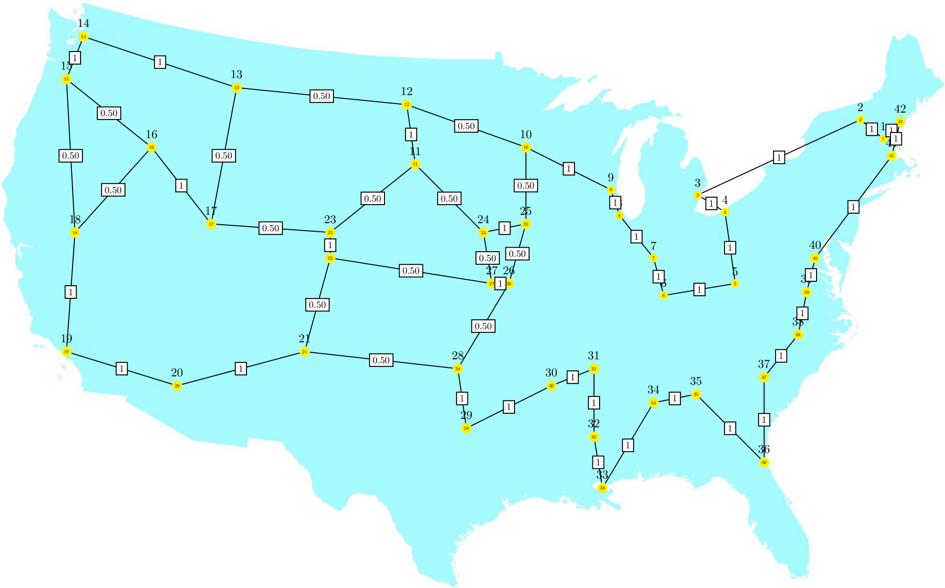
↓ Eliminate Subtour $11 - 23$

8: LP solution 697

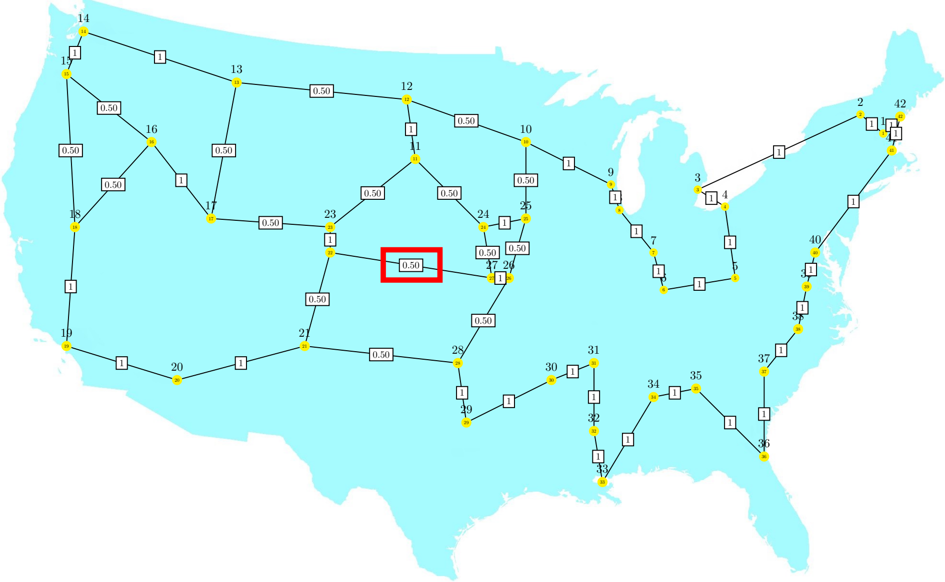
Solving Progress (Alternative Branch 2)



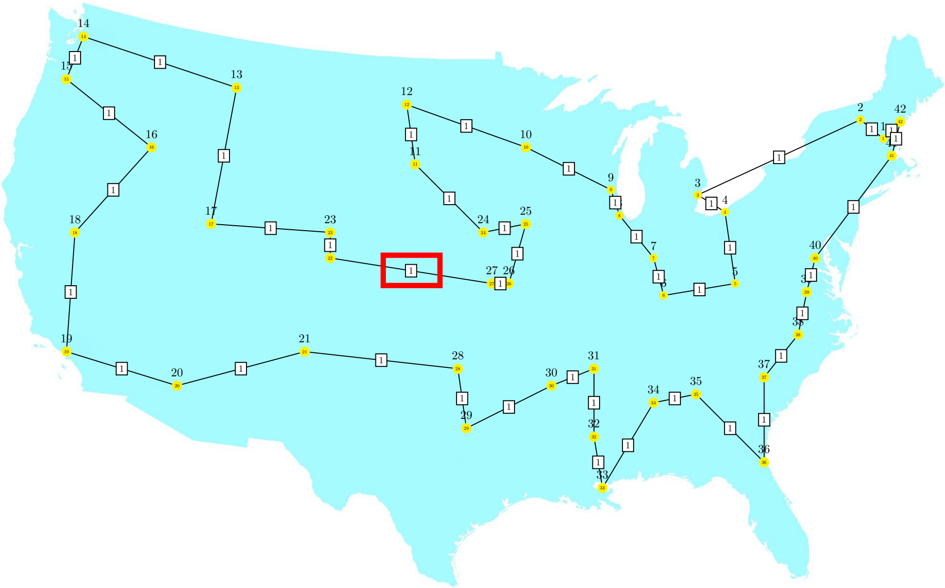
Alternative Branch 2: $x_{27,22}$, Objective 697



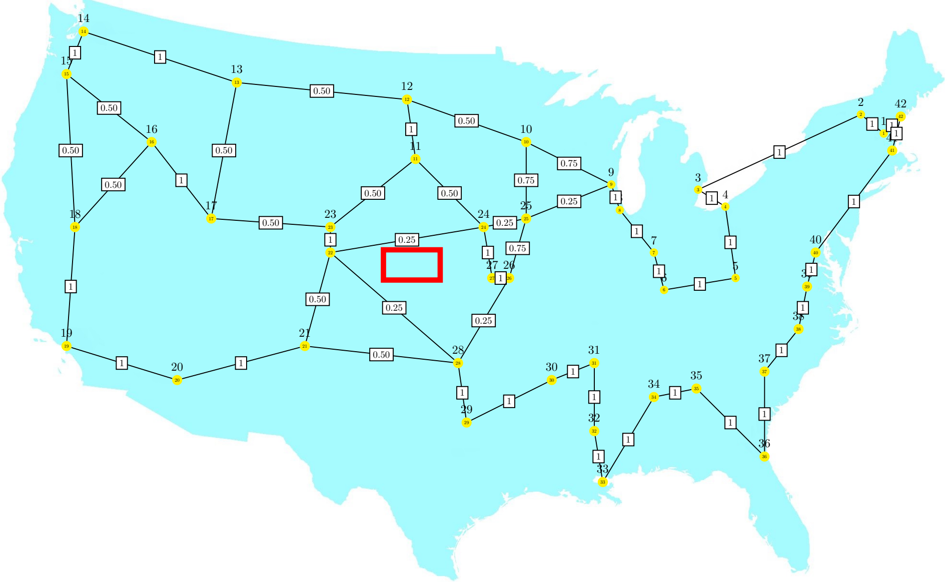
Alternative Branch 2: $x_{27,22}$, Objective 697



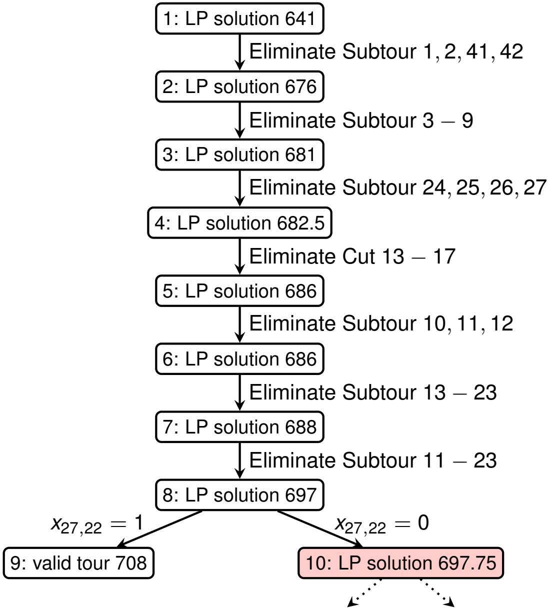
Alternative Branch 2a: $x_{27,22} = 1$, Objective 708 (Valid tour)



Alternative Branch 2b: $x_{27,22} = 0$, Objective 697.75



Solving Progress (Alternative Branch 2)



Solving Progress (Alternative Branch 3)

1: LP solution 641

↓ Eliminate Subtour $1, 2, 41, 42$

2: LP solution 676

↓ Eliminate Subtour $3 - 9$

3: LP solution 681

↓ Eliminate Subtour $24, 25, 26, 27$

4: LP solution 682.5

↓ Eliminate Cut $13 - 17$

5: LP solution 686

↓ Eliminate Subtour $10, 11, 12$

6: LP solution 686

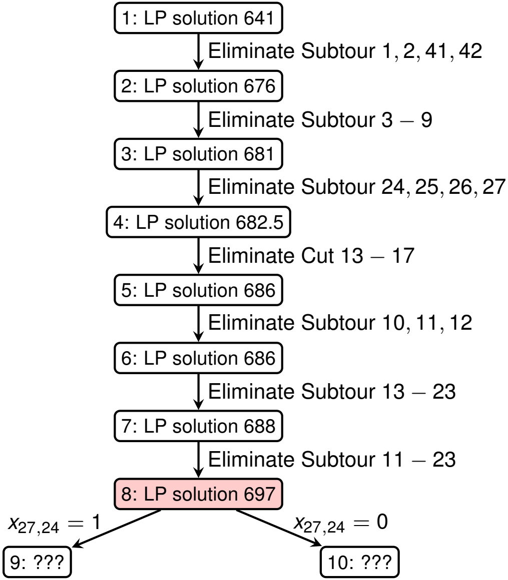
↓ Eliminate Subtour $13 - 23$

7: LP solution 688

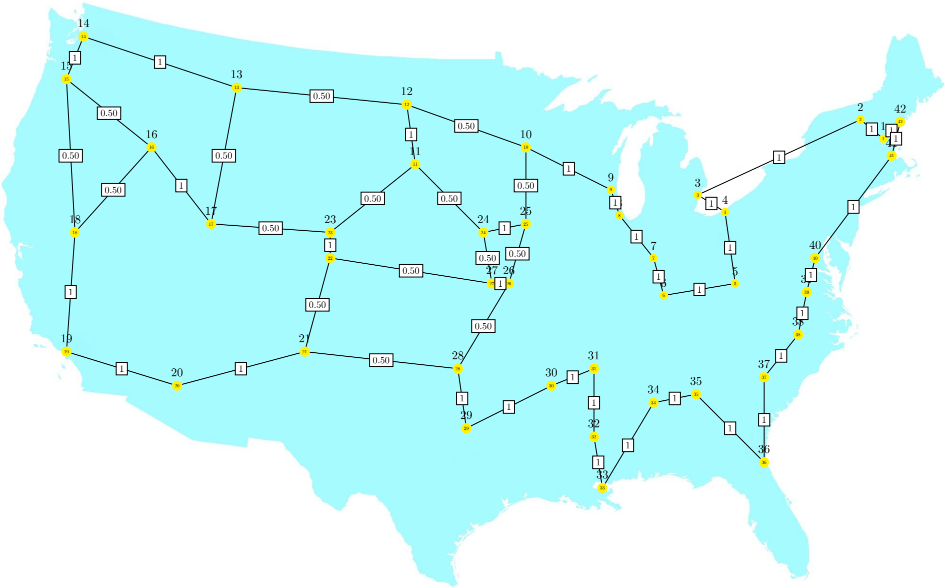
↓ Eliminate Subtour $11 - 23$

8: LP solution 697

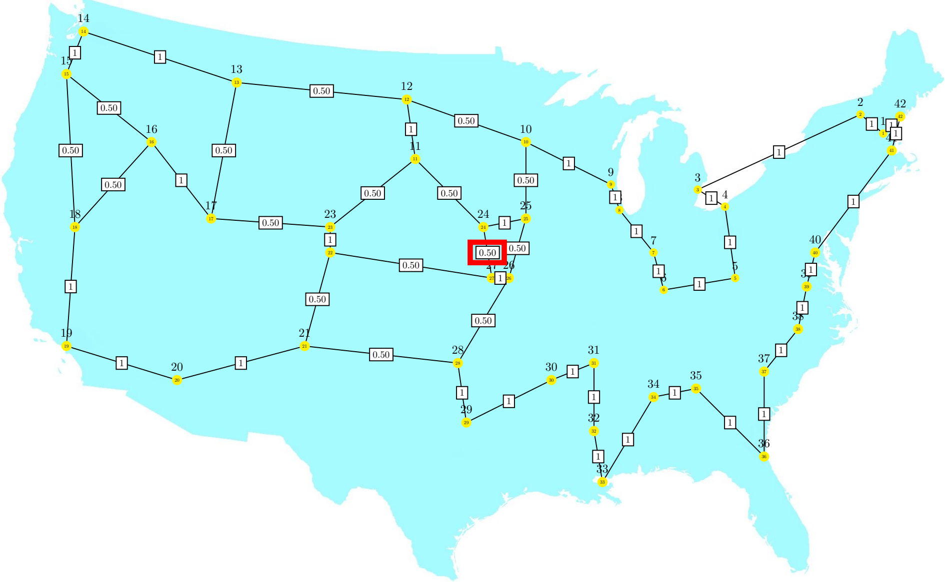
Solving Progress (Alternative Branch 3)



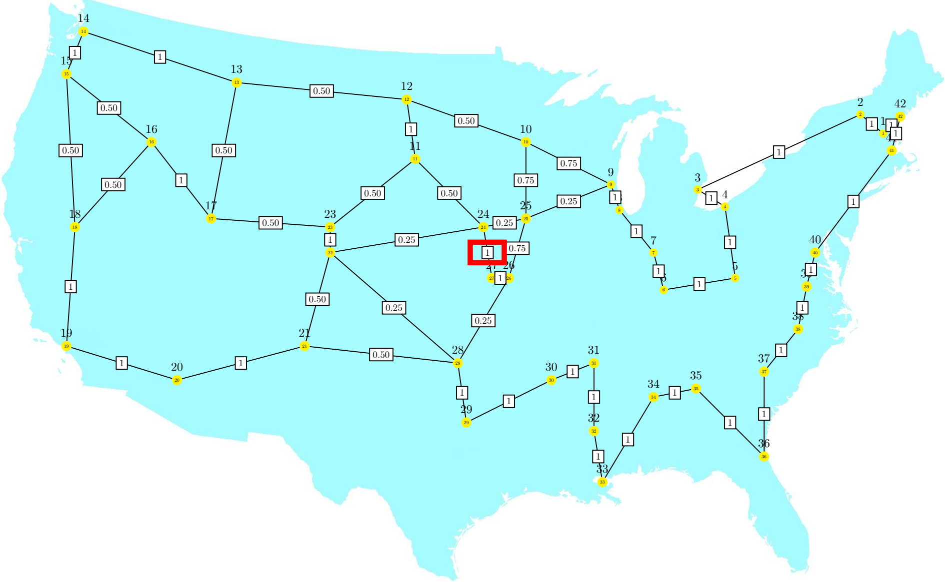
Alternative Branch 3: $x_{27,24}$, Objective 697



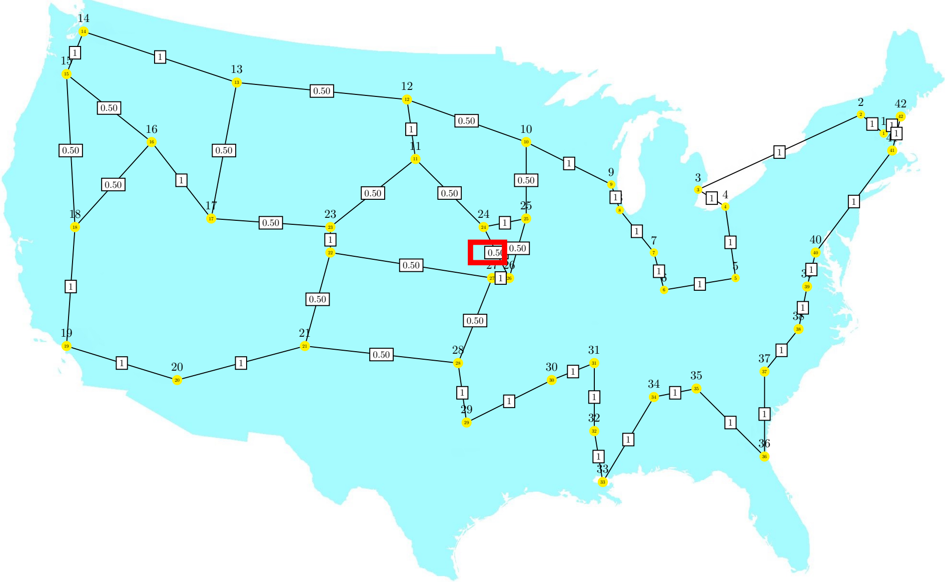
Alternative Branch 3: $x_{27,24}$, Objective 697



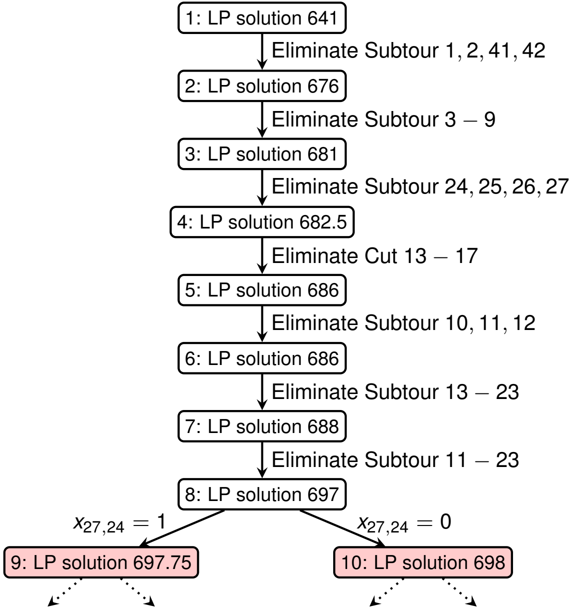
Alternative Branch 3a: $x_{27,24} = 1$, Objective 697.75



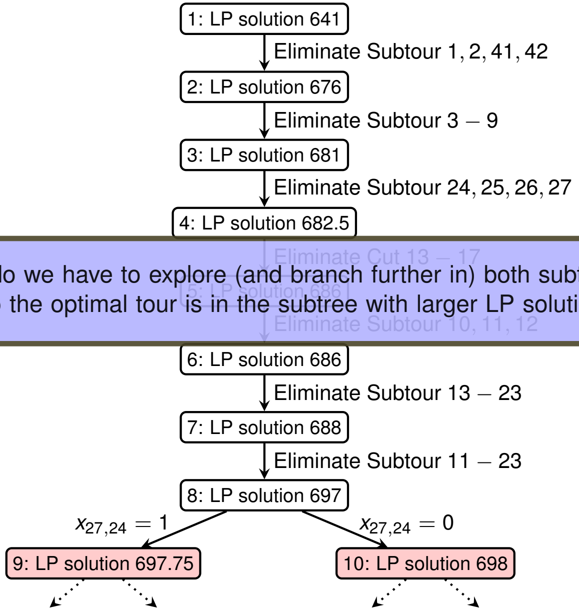
Alternative Branch 3b: $x_{27,24} = 0$, Objective 698



Solving Progress (Alternative Branch 3)



Solving Progress (Alternative Branch 3)



Not only do we have to explore (and branch further in) both subtrees, but also the optimal tour is in the subtree with larger LP solution!

Conclusion (1/2)

- How can one generate these constraints automatically?

Conclusion (1/2)

- How can one generate these constraints automatically?
  Subtour Elimination: Finding Connected Components
  Small Cuts: Finding the Minimum Cut in Weighted Graphs

Conclusion (1/2)

- How can one generate these constraints automatically?
  Subtour Elimination: Finding Connected Components
  Small Cuts: Finding the Minimum Cut in Weighted Graphs

- Why don't we add all possible Subtour Eliminiation constraints to the LP?

Conclusion (1/2)

- How can one generate these constraints automatically?
  Subtour Elimination: Finding Connected Components
  Small Cuts: Finding the Minimum Cut in Weighted Graphs

- Why don't we add all possible Subtour Eliminiation constraints to the LP?
  There are exponentially many of them!

Conclusion (1/2)

- How can one generate these constraints automatically?
  Subtour Elimination: Finding Connected Components
  Small Cuts: Finding the Minimum Cut in Weighted Graphs

- Why don't we add all possible Subtour Eliminiation constraints to the LP?
  There are exponentially many of them!

- Should the search tree be explored by BFS or DFS?

Conclusion (1/2)

- How can one generate these constraints automatically?
  Subtour Elimination: Finding Connected Components
  Small Cuts: Finding the Minimum Cut in Weighted Graphs

- Why don't we add all possible Subtour Eliminiation constraints to the LP?
  There are exponentially many of them!

- Should the search tree be explored by BFS or DFS?
  BFS may be more attractive, even though it might need more memory.

Conclusion (1/2)

- How can one generate these constraints automatically?
  Subtour Elimination: Finding Connected Components
  Small Cuts: Finding the Minimum Cut in Weighted Graphs

- Why don't we add all possible Subtour Eliminiation constraints to the LP?
  There are exponentially many of them!

- Should the search tree be explored by BFS or DFS?
  BFS may be more attractive, even though it might need more memory.

CONCLUDING REMARK

It is clear that we have left unanswered practically any question one might pose of a theoretical nature concerning the traveling-salesman problem; however, we hope that the feasibility of attacking problems involving a moderate number of points has been successfully demonstrated, and that perhaps some of the ideas can be used in problems of similar nature.

Conclusion (2/2)

- Eliminate Subtour $1, 2, 41, 42$
- Eliminate Subtour $3 - 9$
- Eliminate Subtour $10, 11, 12$
- Eliminate Subtour $11 - 23$
- Eliminate Subtour $13 - 23$
- Eliminate Cut $13 - 17$
- Eliminate Subtour $24, 25, 26, 27$

Conclusion (2/2)

- Eliminate Subtour $1, 2, 41, 42$
- Eliminate Subtour $3-9$
- Eliminate Subtour $10, 11, 12$
- Eliminate Subtour $11-23$
- Eliminate Subtour $13-23$
- Eliminate Cut $13-17$
- Eliminate Subtour $24, 25, 26, 27$

THE 49-CITY PROBLEM*

The optimal tour $\bar{x}$ is shown in Fig. 16. The proof that it is optimal is given in Fig. 17. To make the correspondence between the latter and its programming problem clear, we will write down in addition to 42 relations in non-negative variables (2), a set of 25 relations which suffice to prove that $D(x)$ is a minimum for $\bar{x}$. We distinguish the following subsets of the 42 cities:

$S_1 = \{1, 2, 41, 42\}$
$S_2 = \{3, 4, \cdots, 9\}$
$S_3 = \{1, 2, \cdots, 9, 29, 30, \cdots, 42\}$
$S_4 = \{11, 12, \cdots, 23\}$
$S_5 = \{13, 14, \cdots, 23\}$
$S_6 = \{13, 14, 15, 16, 17\}$
$S_7 = \{24, 25, 26, 27\}$.

CPLEX

en.wikipedia.org/wiki/CPLEX

WIKIPEDIA
The Free Encyclopedia

- Main page
- Contents
- Featured content
- Current events
- Random article
- Donate to Wikipedia
- Wikipedia store

Interaction

- Help
- About Wikipedia
- Community portal
- Recent changes
- Contact page

Tools

- What links here
- Related changes
- Upload file
- Special pages

CPLEX

From Wikipedia, the free encyclopedia

IBM ILOG CPLEX Optimization Studio (often informally referred to as simply as CPLEX) is an optimization software package. In 2004, the work on CPLEX earned the first INFORMS Impact Prize.

The CPLEX Optimizer was named for the simplex method as implemented in the C programming language, although today it also supports other types of mathematical optimization and offers interfaces other than just C. It was originally developed by Robert E. Bixby and was offered commercially starting in 1988 by CPLEX Optimization Inc., which was acquired by ILOG in 1997; ILOG was subsequently acquired by IBM in January 2009.[1] CPLEX continues to be actively developed under IBM.

The IBM ILOG CPLEX Optimizer solves integer programming problems, very large[2] linear programming problems using either primal or dual variants of the simplex method or the barrier interior

CPLEX

Developer(s)	IBM
Stable release	12.6
Development status	Active
Type	Technical computing
License	Proprietary
Website	ibm.com/software/products/ibmilogcpleoptistud/

```
Welcome to IBM(R) ILOG(R) CPLEX(R) Interactive Optimizer 12.6.1.0
  with Simplex, Mixed Integer & Barrier Optimizers
5725-A06 5725-A29 5724-Y48 5724-Y49 5724-Y54 5724-Y55 5655-Y21
Copyright IBM Corp. 1988, 2014.  All Rights Reserved.

Type 'help' for a list of available commands.
Type 'help' followed by a command name for more
information on commands.

CPLEX> read tsp.lp
Problem 'tsp.lp' read.
Read time = 0.00 sec. (0.06 ticks)
CPLEX> primopt
Tried aggregator 1 time.
LP Presolve eliminated 1 rows and 1 columns.
Reduced LP has 49 rows, 860 columns, and 2483 nonzeros.
Presolve time = 0.00 sec. (0.36 ticks)

Iteration log . . .
Iteration:     1    Infeasibility =             33.999999
Iteration:    26    Objective     =           1510.000000
Iteration:    90    Objective     =            923.000000
Iteration:   155    Objective     =            711.000000

Primal simplex - Optimal:  Objective =  6.9900000000e+02
Solution time =    0.00 sec.  Iterations = 168 (25)
Deterministic time = 1.16 ticks  (288.86 ticks/sec)

CPLEX> 
```

```
CPLEX> display solution variables -
Variable Name           Solution Value
x_2_1                         1.000000
x_42_1                        1.000000
x_3_2                         1.000000
x_4_3                         1.000000
x_5_4                         1.000000
x_6_5                         1.000000
x_7_6                         1.000000
x_8_7                         1.000000
x_9_8                         1.000000
x_10_9                        1.000000
x_11_10                       1.000000
x_12_11                       1.000000
x_13_12                       1.000000
x_14_13                       1.000000
x_15_14                       1.000000
x_16_15                       1.000000
x_17_16                       1.000000
x_18_17                       1.000000
x_19_18                       1.000000
x_20_19                       1.000000
x_21_20                       1.000000
x_22_21                       1.000000
x_23_22                       1.000000
x_24_23                       1.000000
x_25_24                       1.000000
x_26_25                       1.000000
x_27_26                       1.000000
x_28_27                       1.000000
x_29_28                       1.000000
x_30_29                       1.000000
x_31_30                       1.000000
x_32_31                       1.000000
x_33_32                       1.000000
x_34_33                       1.000000
x_35_34                       1.000000
x_36_35                       1.000000
x_37_36                       1.000000
x_38_37                       1.000000
x_39_38                       1.000000
x_40_39                       1.000000
x_41_40                       1.000000
x_42_41                       1.000000
All other variables in the range 1-861 are 0.
```