

Exactly solving TSP using the Simplex algorithm

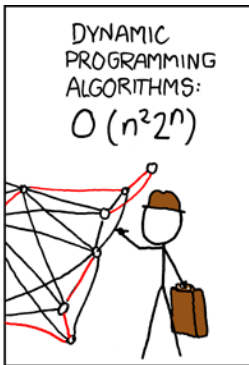
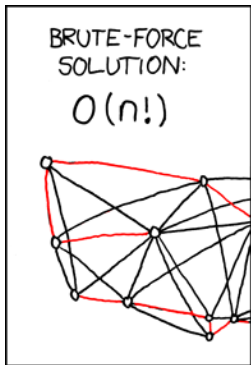
Andrej Ivašković, Thomas Sauerwald

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: $dp(1, \{1\}) = 0$.
- ▶ Recurrence relation:

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

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:

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

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, $O(n 2^n)$ 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:

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

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

Demo: final solution

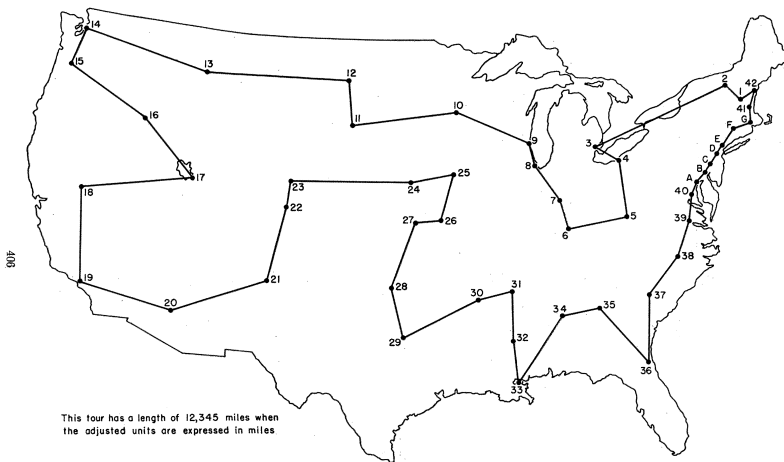


Fig. 16. The optimal tour of 49 cities.

Demo: materials

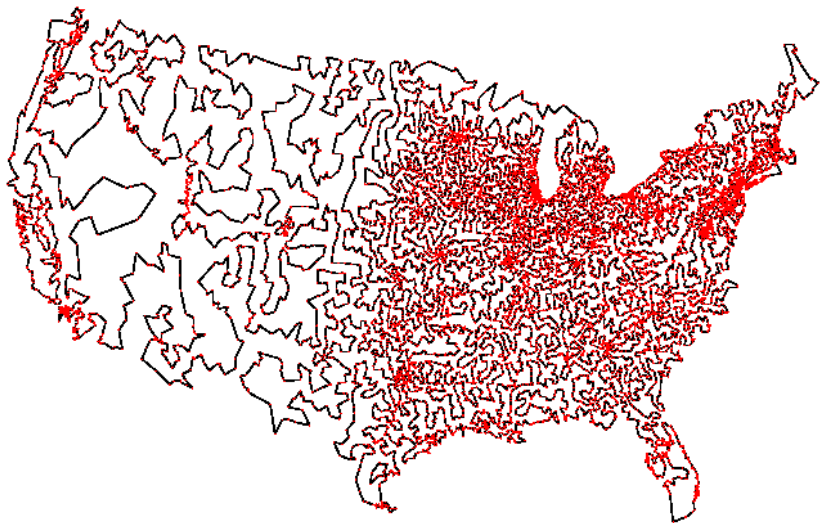
- ▶ The full implementation of this TSP solver in C++ (along with all the necessary files to perform this demo) may be found at:

<https://github.com/PetarV-/Simplex-TSP-Solver>

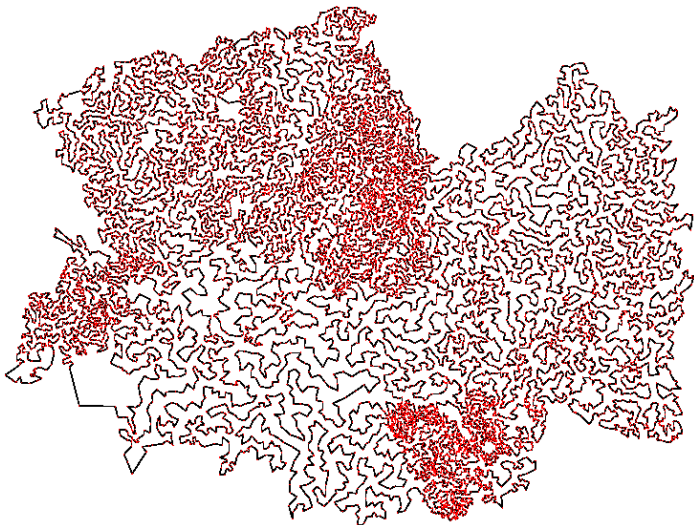
- ▶ Methods similar to these have been successfully applied for solving far larger TSP instances. For example:

<http://www.math.uwaterloo.ca/tsp/>

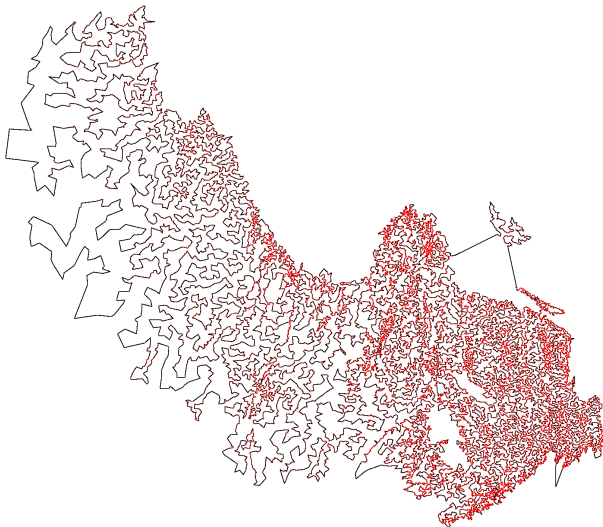
13,509 largest towns in the US



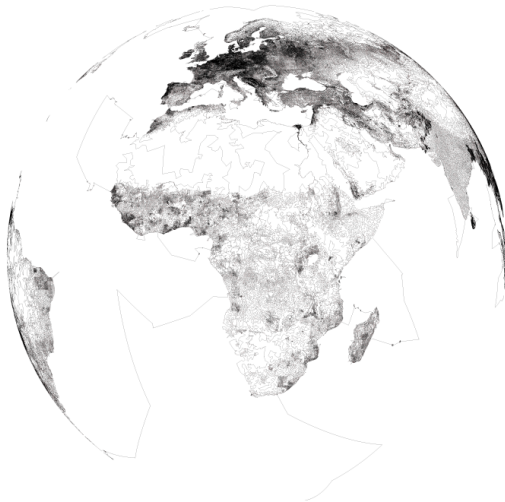
15,112 largest towns in Germany

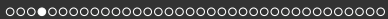


All 24,978 populated places in Sweden

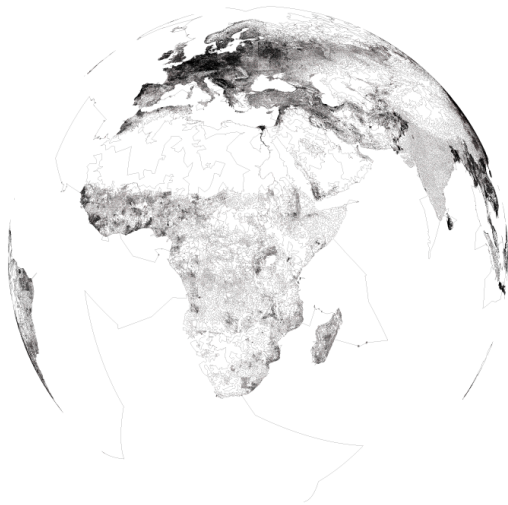


World TSP – 1,904,711 places, error $\leq 0.05\%$

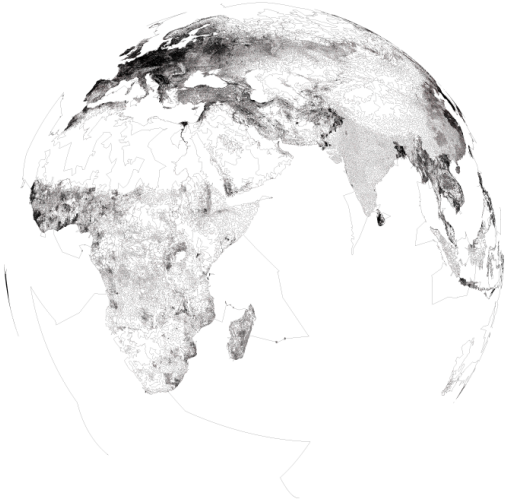




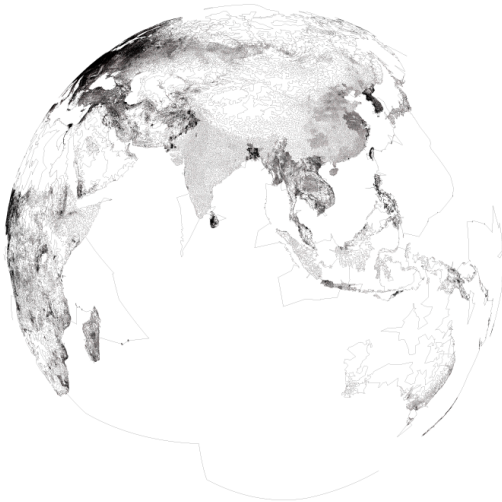
World TSP – 1,904,711 places, error $\leq 0.05\%$



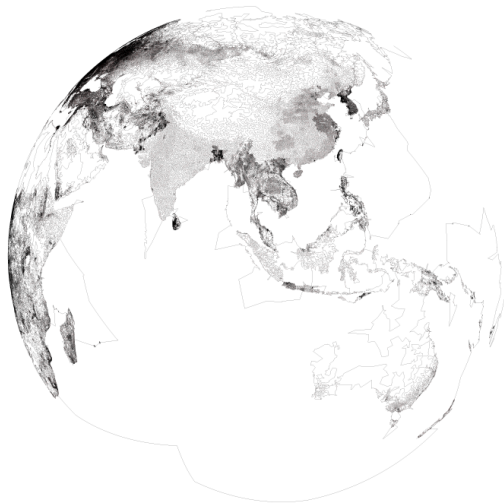
World TSP – 1,904,711 places, error $\leq 0.05\%$



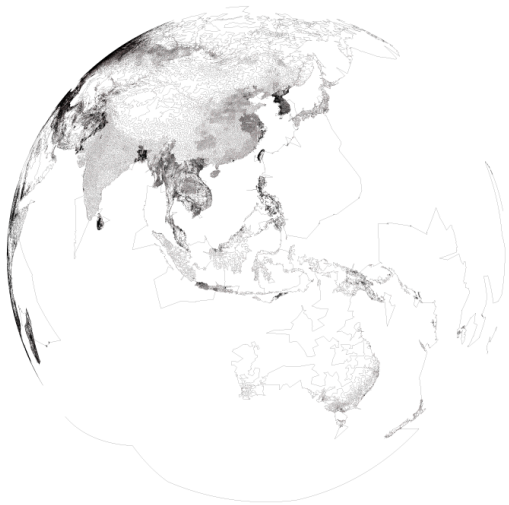
World TSP – 1,904,711 places, error $\leq 0.05\%$



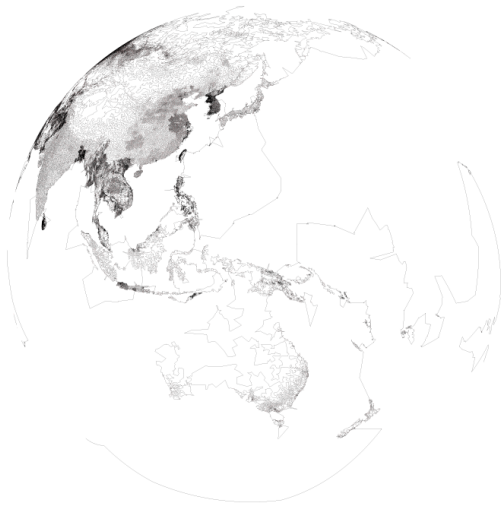
World TSP – 1,904,711 places, error $\leq 0.05\%$



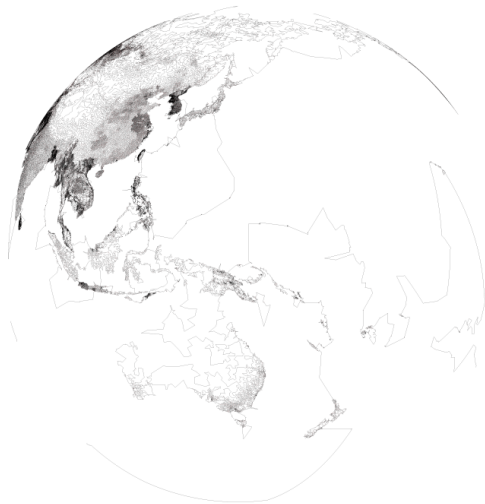
World TSP – 1,904,711 places, error $\leq 0.05\%$



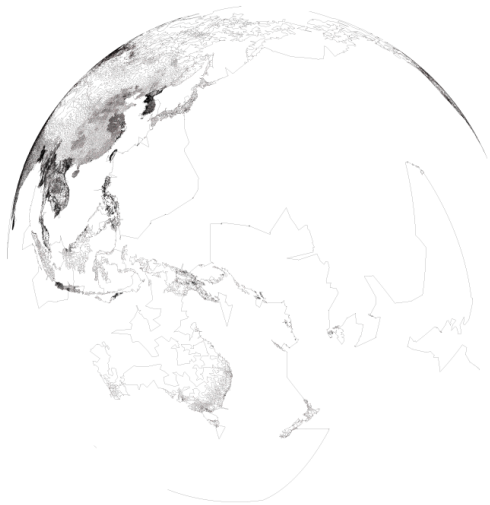
World TSP – 1,904,711 places, error $\leq 0.05\%$



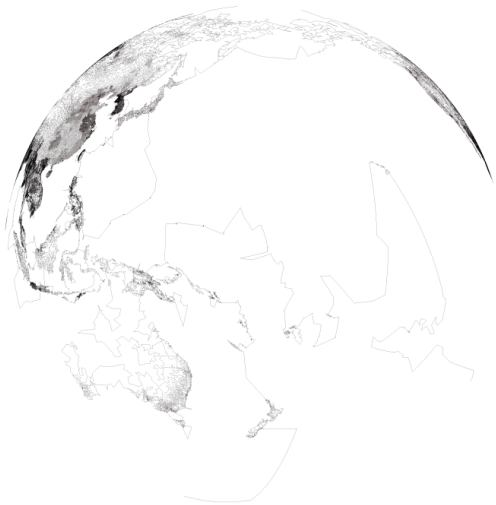
World TSP – 1,904,711 places, error $\leq 0.05\%$



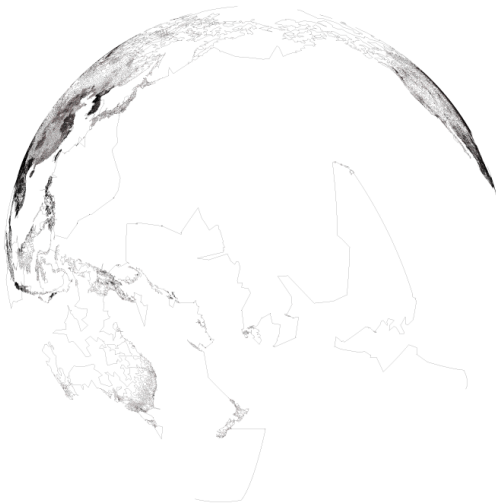
World TSP – 1,904,711 places, error $\leq 0.05\%$



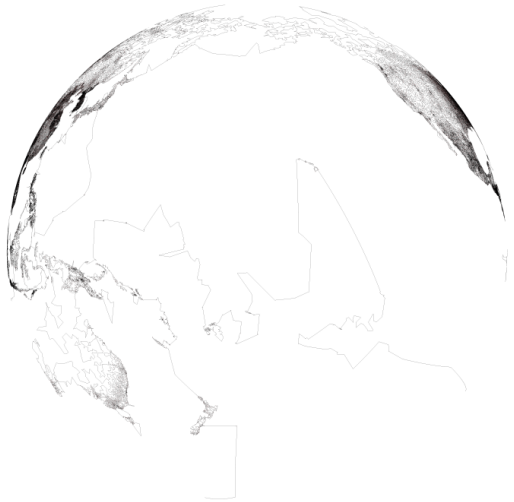
World TSP – 1,904,711 places, error $\leq 0.05\%$



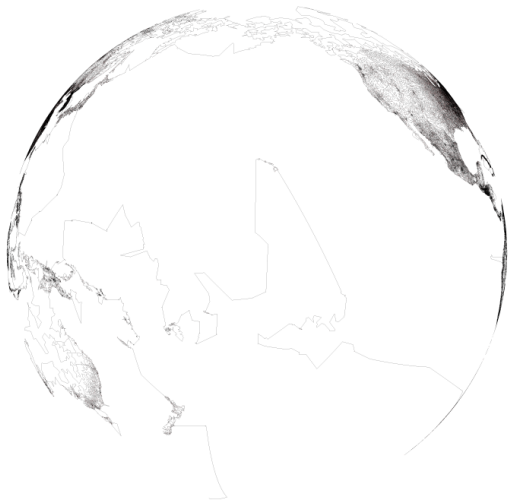
World TSP – 1,904,711 places, error $\leq 0.05\%$



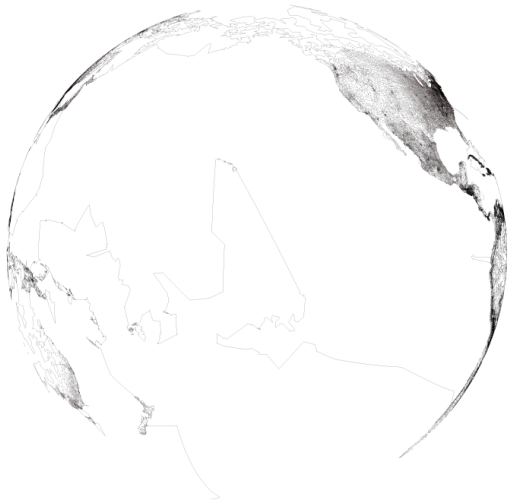
World TSP – 1,904,711 places, error $\leq 0.05\%$



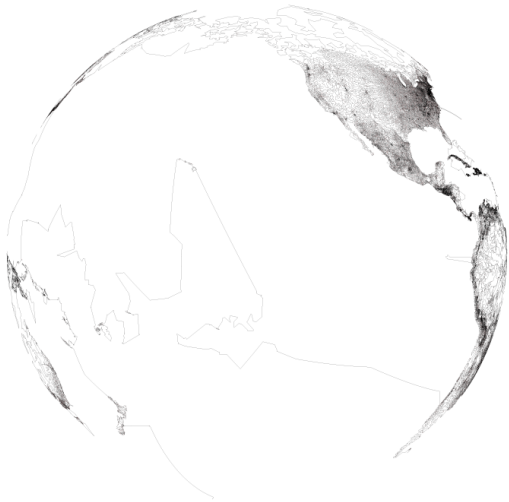
World TSP – 1,904,711 places, error $\leq 0.05\%$



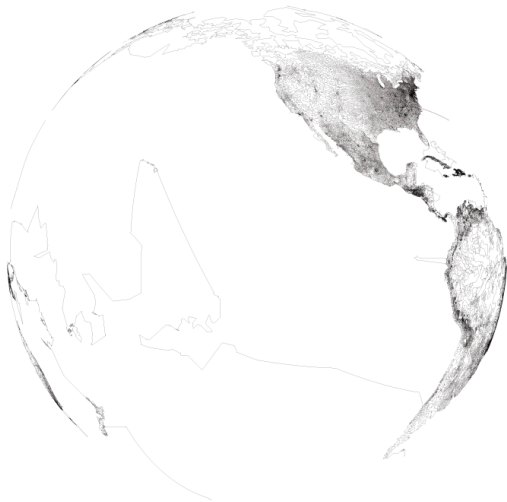
World TSP – 1,904,711 places, error $\leq 0.05\%$



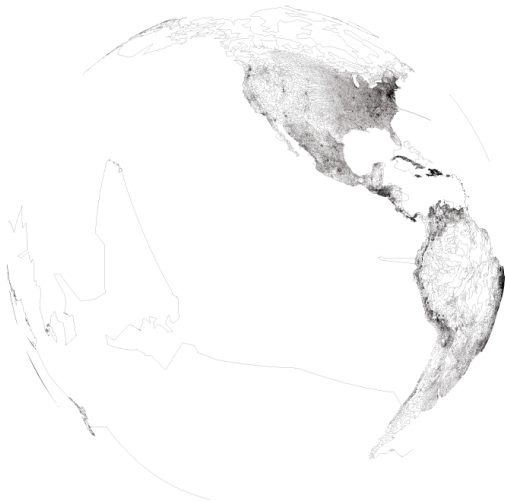
World TSP – 1,904,711 places, error $\leq 0.05\%$



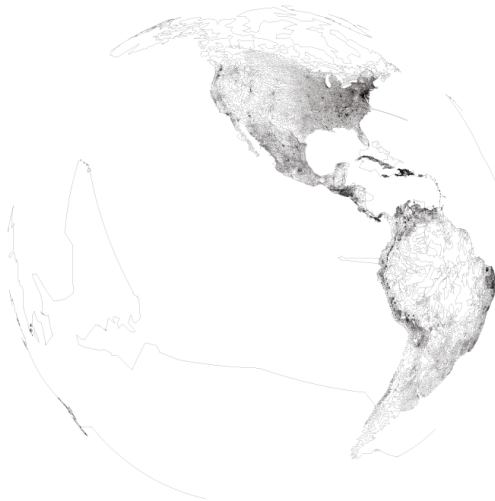
World TSP – 1,904,711 places, error $\leq 0.05\%$



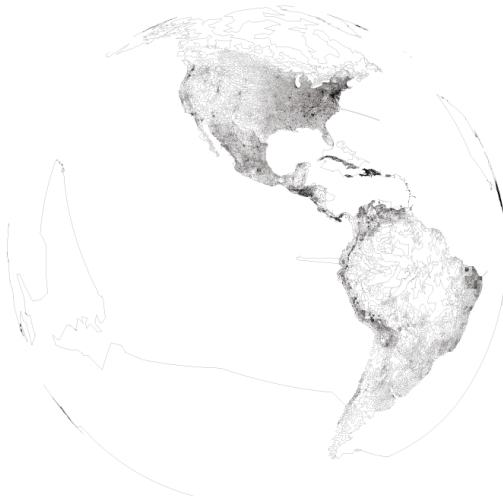
World TSP – 1,904,711 places, error $\leq 0.05\%$



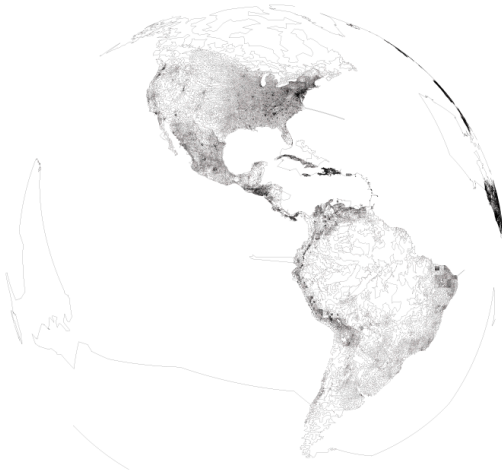
World TSP – 1,904,711 places, error $\leq 0.05\%$



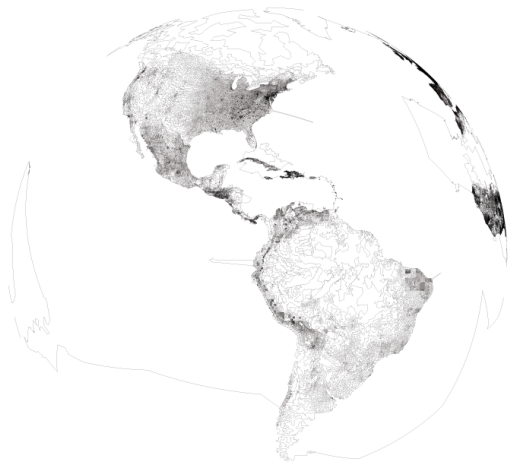
World TSP – 1,904,711 places, error $\leq 0.05\%$



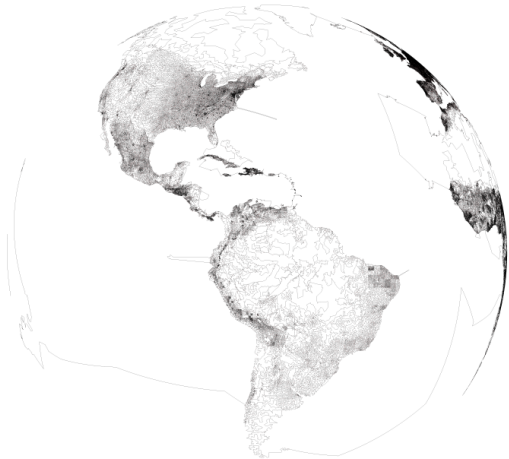
World TSP – 1,904,711 places, error $\leq 0.05\%$



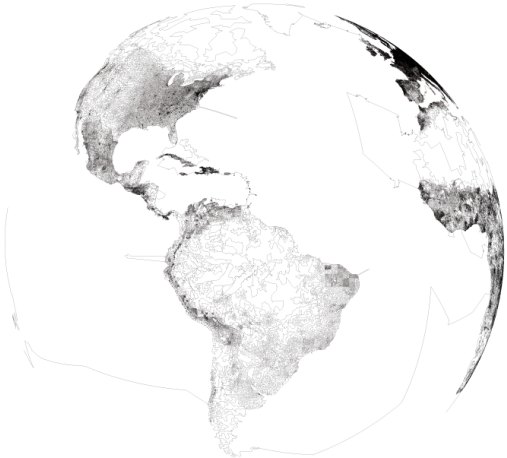
World TSP – 1,904,711 places, error $\leq 0.05\%$



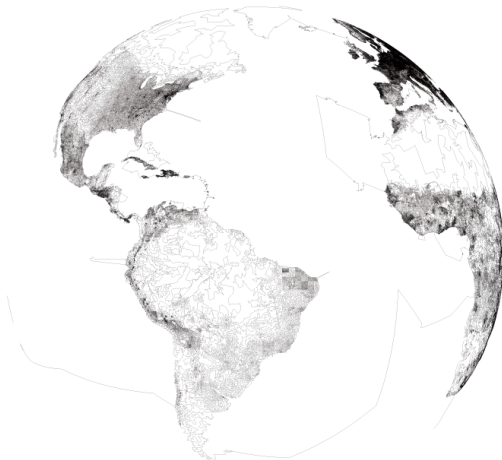
World TSP – 1,904,711 places, error $\leq 0.05\%$



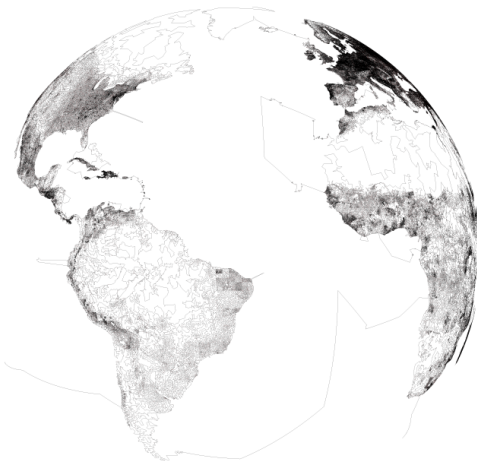
World TSP – 1,904,711 places, error $\leq 0.05\%$



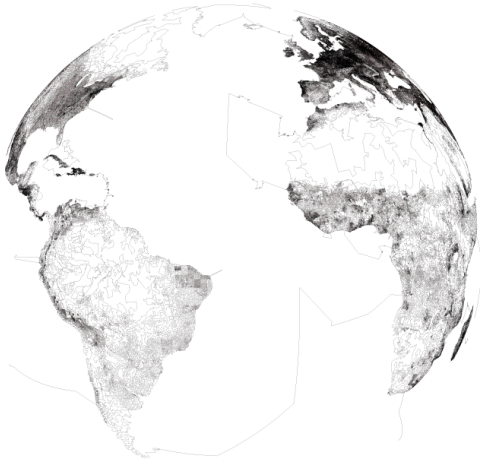
World TSP – 1,904,711 places, error $\leq 0.05\%$



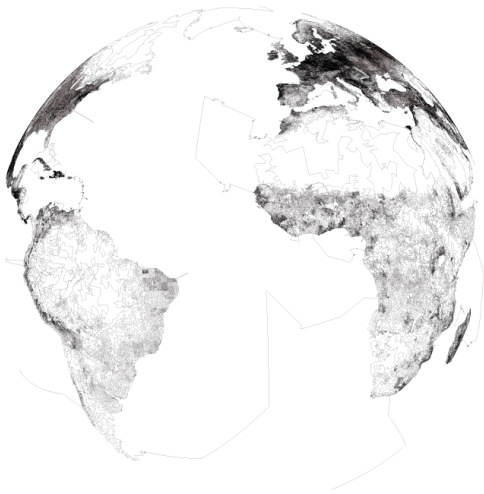
World TSP – 1,904,711 places, error $\leq 0.05\%$



World TSP – 1,904,711 places, error $\leq 0.05\%$



World TSP – 1,904,711 places, error $\leq 0.05\%$



World TSP – 1,904,711 places, error $\leq 0.05\%$

