Excel Solvers for the Traveling Salesman Problem

Mangesh Gharote, Dilys Thomas, Sachin Lodha mangesh.g@tcs.com dilys@cs.stanford.edu sachin.lodha@tcs.com Tata Consultancy Services, Pune, India

ABSTRACT

Ordering queries within a workload and ordering joins in a query are important problems in databases [1]. We give algorithms for the query sequencing problem that scale (small space) and are efficient (low runtime) as compared to earlier work [4]. The errors are small in practice and we are able to further reduce them using geometric repair. We provide a computational comparison of TSP solvers and show extensive testing on benchmark datasets [25] observing its connection to these ordering problems.

1. PROBLEM STATEMENT

Database systems are facing an ever increasing demand for high performance. Either as standalone Oracle, SQLServer or DB2 installations or as a backend to Peoplesoft, SAP or Siebel workloads they are required to execute a batch of queries that contain several common subexpressions. Traditionally, query optimizers like [37], [36] optimize queries one at a time and do not identify any commonalities in queries, resulting in repeated computations. As observed in [3, 39] exploiting common results, multi-query optimization (MQO), can lead to significant performance gains – this requires the queries to be ordered in the workload for memory reuse and reduced disk need. Motivated by the importance for ordering problems, we study the combinatorial ordering problem of the travelling salesman problem (TSP) and provide extensive testing on benchmark datasets [25].

1.1 Applications

The traveling salesman problem has wide applicability in many different industrial and scientific scenarios. Some notable ones are: vehicle routing, bus scheduling, development of flight schedules, crew scheduling, order-picking problem in warehouses, printing press scheduling problem, network cabling in a country, computer wiring, query workload ordering for optimization, VLSI chip design connectivity layout, drilling of printed circuit boards, genome sequencing, hot rolling scheduling problem in iron & steel industry, overhauling gas turbine engines, X-Ray crystallography (ordering positions for measurement), global navigation satellite system, ordering test cases in regression suite to re-use components etc.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

The 18th International Conference on Management of Data (COMAD), 14th-16th Dec, 2012 at Pune, India.

Copyright © 2012 Computer Society of India (CSI).

See [6] for a description of some applications of TSP. Intractability [12] [11] and restricted tractability results [9] [10] for TSP have won top awards. We develop our own algorithms on top of reasonable in-practice TSP algorithms. We obtain near optimal tours in practice. Our aim is to reduce run time and be scalable in memory for medium to large instances of TSP. Ease of using the tool, ability to handle different distance metrics including longitude and latitude, and ease of visualizing the tours produced are the aims of our project of improving state of the art TSP solvers available in Excel [4].

2. NEAREST NEIGHBOR AND GREEDY AL-GORITHMS

2.1 Nearest Neighbor

Algorithm 1 implemented in our Excel solver is the Nearest Neighbor(NN) algorithm. Since it grows a single segment, it is similar to left deep plans used in query optimizers. Different start points can give different tours, see Figure 1.

Algorithm 1 Nearest Neighbor

Select an arbitrary vertex as current vertex.

while not all the vertices in domain are visited do

Find shortest edge connecting current vertex and an unvisited vertex V.

Set current vertex to V. Mark V visited.

end while

2.2 Greedy

Instead of starting from one vertex in NN, Algorithm 2 the greedy algorithm grows multiple segments and stitches them together to get a tour, similar to bushy optimizer plans.

Algorithm 2 Greedy

Sort all edges.

while less than n edges in tour do

Select the shortest edge and add it to tour if

- [1] not yet on tour and not creating a degree-3 vertex.
- [2] not creating a cycle of size less than n.

end while

3. TOUR REPAIR

NN cannot approximate TSP to better than a factor of log(n) [40] and may produce the worst possible tour [13]. In practice NN and

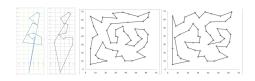


Figure 1: Different start points in 16 NN(32% from opt),(5% from opt), 51 NN, Greedy(intersection removal, section 3.1)(8% from opt),(11% from opt)

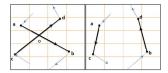


Figure 2: Intersection Unrolling

greedy gives within 25% away from optimal for moderately large sized instances. See Figures 6, 7, 8. The solutions obtained can be further repaired with our intersection removal, hinge-crest optimization, and tested techniques like geometric constructions, k-opt, etc.

3.1 Intersection Unrolling

From Figure 2 (i) Triangle Inequality ao + co > ac. (ii) Again do + ob > db. (iii) Adding (i) & (ii) ao + co + do + bo > ac + db. (iv) Rearranging terms ao + ob + co + do > ac + db. (v) Intersection Unrolling ab + cd > ac + db. We solve for the intersection point using Cramers Rule. Intersection unrolling is applied when intersection point lies on both segments, as shown in Algorithm 3. For every i, j intersection, the tour between vertices Tour[i+1] and Tour[j] has been reversed by the inner while. See Figure 3 for

Algorithm 3 Unroll Intersection

examples.

3.2 Hinge and Crest Optimization

The hinge and crest optimization (transfer tour repair) from Figure 4 is given in Algorithm 4 and applied in Figure 5.

4. RELATED WORK AND EXPERIMENTS

Being the most important geometric combinatorial problem, the TSP has multiple popular algorithms.

4.1 Lin-Kernighan

Lin-Kernighan heuristic tries removing k edges and adding k other edges aiming to retain a tour but to reduce the cost taking at most $O(n^k)$ time.

4.2 Linear Programming Formulation, Cutting Plane



Figure 3: Intersection Removal on 16 NN(5% from opt),(3% from opt) and 16 Greedy(17% from opt),(1% from opt)

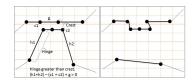


Figure 4: Hinge and Crest Transfer



Figure 5: Hinge and Crest Transfer, 51 points

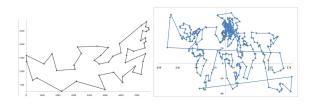


Figure 6: 48 US mainland capitals(our 7%),6 continents 535 airports(10%)

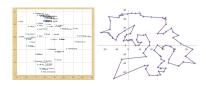


Figure 7: India 67 cities(our 1%), Africa and Islands(our 12%)



Figure 8: 2103 points PCB drilling(6%), Converting Pictures to Tours using Voronoi diagrams [33](2)

Algorithm 4 Transfer tour repair

end while

```
while there exists nearby points on different segments do if hinge distance > crest distance i.e. h_1 + h_2 + g_1 - g_2 - c_1 - c_2 > 0 then

Transfer points to nearer segment and decrease cost. end if
```

Algorithm 5 Computing Dij from longitude and latitude [25]

```
PI = 3.141592. R=6378.388. /* Radius of earth*/
degree = (int) X[i]. minute = X[i] - degree.
radian = PI * (degree + 5 *minute/3)/180.
v1 = cos(lng[i] -lng[j]).
v2 = cos(lat[i] - lat[j]). v3 = cos(lat[i] + lat[j]).
Dij = (int) (R * acos(1/2 *((1 + v1)*v2 - (1 - v1)*v3))+1).
```

Miller-Tucker-Zemlin were among the first to provide formulations for TSP [14].

```
\begin{array}{l} \min \sum_{i \in V} \sum_{j \in V, j > i} c_{ij} y_{ij} \text{ (minimize tour cost), Subject to,} \\ \sum_{j \in V, j > i} y_{ij} + \sum_{j \in V, j < i} y_{ji} = 2 \ \forall i \in V \text{ (vertex degree two),} \\ \sum_{i \in S} \sum_{j \in S, j > i} y_{ij} \leq |S| - 1 \ \forall \phi \neq S \subset V \text{ (no subtours),} \\ 0 \leq y_{ij} \leq 1, \forall i, j \in V, j > i, y_{ij} \text{ integer } \forall i, j \in V, j > i. \end{array}
```

We use the bounds obtained from the Held Karp lower bound [17, 18, 28], an LP relaxation, in Table 1 (see [25]). [4] uses in its backend linear programming solvers like CPLEX, Gurobi, Xpress solvers for solving the TSP problem.

Concorde solver developed by Robert Bixby, Vasek Chvatal, William Cook and David Applegate [7, 8], uses the cutting plane technique.

4.3 Held Karp Dynamic Programming

Algorithm 6, Held-Karp [15] dynamic programming is a (n^22^n) time complexity algorithm for TSP. This memoizes the solutions to 2^n subsets of locations. Take some starting vertex s for the tour. For set of vertices R, $s \in R$, vertex $w \in R$, let $B(R, w) = \min$ imum length of a path, starting in s visiting only all vertices in R and ending in w. Remembering the optimal subsolution (dynamic

Algorithm 6 Held Karp

```
\begin{array}{l} B(\{s\},s)=0.\\ \text{for all }S \text{ and }w \text{ and }|S|>1 \text{ do}\\ B(S,w)=min_{v\in S-\{w\}}B(S-\{w\},v)+weight(v,w).\\ \text{end for} \end{array}
```

programming) for subsets reduces exponential term of the running time from n! $((n/e)^n)$ to 2^n . It is a 50 year open problem if there is an exact algorithm for TSP with time (c^n) for c < 2 [27] (some recent progress has been made for cubic graphs [21, 20] and hamiltonian paths [19]). Memoization is popular in modern query optimizers including map reduce contexts [38].

4.4 Christofides

Algorithm 7, Christofides's algorithm [16] is a 1.5 approximation to metric TSP. The MST (minimum spanning tree) is atmost the cost of $1 \times TSP$ as a TSP tour without a single edge is a spanning tree. A min weight matching is atmost $0.5 \times TSP$ as odd / even edges in a TSP tour give a matching. In practice 10-20% away from optimal solutions have been obtained [26]. It is a 35 year open problem if there is an approximation algorithm with factor < 1.5 (some recent progress has been made at Stanford for shortest path graph metrics [22, 23]). For the asymmetric case a similar algorithm recently developed by our colleagues at Stanford University

Algorithm 7 Christofides

Get a MST T using Prim's or Kruskal's algorithm. Set $O = \{v \mid v \text{ has odd degree in tree } T\}$. Compute a minimum weight matching M in the graph G[O]. Compute Euler tour C in graph T union M. Add shortcuts to C to get a TSP-tour.

size	nn	nn-int	greedy	greedy-int
14	15.6	13.6	17	16.6
16	5.4	2.8	17.6	1.0
48	13	7.1	19.7	11.7
51	19.2	8.5	13	11
52	8.5	3.5	32.0	24.1
67	7.2	1.2	18.2	1
96	18.4	12.1	20.6	16.5
101	17	11.1	26.3	24.2
280	21.4	12.5	14.8	8.1
535	20.7	19.3	15.4	10.1
783	25	16.4	19.6	12.6
1002	21.4	13.6	19.2	14.4
2103	9.4	6.5		
14051	21.3	13.8		
33180	19.1	12.6		
85900	15.2	10.1		

Table 1: Performance of Excel Solver- %age away from optimal

achieves O(log n / loglog n) approximation [5].

4.5 Tours and Rectifications

Starting from size 33 instance in 1950s, the largest instance solved optimally till date is 85,900 locations taking 136 CPU years. Our results from Table 1 (for datasets from [25] except 67 in Figure 7) gives the percentage difference from optimal (obtained from Held Karp lower bound and [25]) of the solutions obtained from NN and greedy algorithms and with the intersection removal algorithm applied to the solutions. Greedy performs better on larger datasets but is more time expensive.

4.6 Metaheuristics

We also experimentally implemented heuristics like Simulated Annealing (SA)[31], Ant Colony Optimization (ACO)[30] and ElectroMagnetism(EM) like algorithm [32] for the TSP Problem whose results are shown in Table 2. Their complicated expensive noncombinatorial iteration rules lead to poor performance in CPU, RAM and approximation ratio especially as instance sizes increase.

size	EM	SA	ACO
14	15.0	18.4	15.0
52	8.5	17.2	6.5
96	18.2	35.9	14.2
159	15.4	29.5	14.3
226	15.9	17.6	13.1
299	20.2	27.9	20.8
654	24.2	28.3	24.0

Table 2: Performance of Metaheuristics- %age away from op-

4.7 SQL Workload

In the first experiment, we generated 5 workloads with 100 queries each, each query a join of a random subset of 20 tables. Distance between two queries (with sets of tables \Re_1 and \Re_2) is the cardinality of the symmetric difference of the sets of tables in each queries join ($|\Re_1 \Delta \Re_2|$). This allows shared pipelined table scans and LRU RAM reuse. On an average across workloads, we observed the schedule developed by NN to be 3.7%, and greedy to be 2.9% away from optimal. In the second experiment we generated 5 workloads with 1000 queries each, each query selecting each table from totally 100 tables with probability 0.2 to be in the query's join (each approximately a 20 table join). On average 9.7 tables were shared between adjacent queries in the optimal ordering. The schedule developed by NN was 3.6% and greedy 2.3% away from optimal on average with 8.8 tables shared between adjacent queries compared to a random ordering that could achieve only four tables shared between adjacent queries. Considering columnar storages and cache policies, in a third experiment we considered a real world SAP workload containing 924 queries which reference on average 7.4 columns per query. The reordering increased the number of columns shared between adjacent queries from 0.42 to 4.9 on average. In a fourth experiment, a real world SAP workload of 16000 queries with on average 13.8 columns per query had originally 1.8 columns shared between adjacent queries already showing affinity, and after reordering shared 13.1 columns between adjacent queries, most being with same prepared statement template groupings. Template groupings make, batch execution techniques like JDBC rewrite [2], and cache reuse techniques [34, 35] that use LRU algorithm and time based aging across foreign keys, possible.

4.8 Critique of work

The most recent excel TSP solver [4] could solve upto 180 cities without running out of memory or time. We present a solver that can solve instances of upto 85,900 cities the largest instance solved optimally to date, approximately. With no extra software installation and a click of a button we are able to solve multiple different large sized TSP problems and provide tour rectifications for ordering problems. We provide an understanding of TSP solvers and show extensive testing on benchmark ordering problem datasets [25]. NEOS solver requires expensive dedicated servers [29].

Acknowledgements: Metaheuristics were developed with help from Prem Nathan, Prashant Kumar and Sani Kumbhar. Dr. Maitreya Natu provided the SAP workload. Dr. Rajiv Raman provided a few recent references.

5. REFERENCES

- S. Sudarshan, A. A. Diwan, Dilys Thomas, Scheduling and caching in multiquery optimization, COMAD 2006, 150–153.
- [2] M. Chavan, R. Guravannavar, K. Ramachandra, S. Sudarshan, DBridge: A program rewrite tool for set-oriented query execution, ICDE 2011.
- [3] P. Roy, S. Seshadri, S. Sudarshan, and S. Bhobhe. Efficient and extensible algorithms for multi-query optimization, SIGMOD 2000, 249–260.
- [4] Rasmus Rasmussen, TSP in spreadsheets: A fast and flexible tool, Elsevier, Omega 39, 1, 51–63, January 2011.
- [5] Arash Asadpour, Michel Goemans, Aleksander Madry, Shayan Oveis Gharan, Amin Saberi, An O(log n / log log n)-approximation algorithm for the asymmetric travelling salesman problem, SODA 2010.
- [6] Donald Davendra, Traveling Salesman Problem, Theory and

- Applications, URL: http://www.intechopen.com, December 2010.
- [7] Vasek Chvatal, Robert Bixby, William Cook, David Applegate, Traveling salesman problem: A computational study, PUP, 2006.
- [8] D. Applegate, R. Bixby, V. Chvatal, and W. Cook, Concorde, TSP Solver, URL: http://www.tsp.gatech.edu/concorde/, 2006
- [9] Sanjeev Arora, Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems, JACM, 1998, 45, 5.
- [10] Mitchell, J. S. B., Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems, SIAM Journal on Computing, 1999.
- [11] Stephen Cook, The complexity of theorem proving procedures, STOC 1971, 151-158.
- [12] Richard Karp, Reducibility among combinatorial problems, Complexity of Computer Computations, 1972, 85-103.
- [13] J. Bang-Jensen, G. Gutin, A. Yeo, When the greedy algorithm fails, Discrete Optimization 1, 2004, 121-127.
- [14] C.E. Miller, A.W. Tucker, R.A. Zemlin, Integer programming formulations and traveling salesman problems, JACM, 7, 1960, 326–329.
- [15] M. Held, R. Karp. A dynamic programming approach to sequencing problems, Journal of SIAM, 1962, 10, 196-210.
- [16] Nicos Christofides, Worst-case analysis of a new heuristic for the traveling salesman problem, Report 388, GSIA, CMU, 1976.
- [17] M. Held, R. M. Karp, The traveling-salesman problem and minimum spanning trees, Operations Res. 18, 1970, 1138-1162.
- [18] M. Held, R. M. Karp, The traveling-salesman problem and minimum spanning trees: Part II, Math. Programming 1, 1971, 6-25.
- [19] Andreas Björklund, Determinant Sums for Undirected Hamiltonicity, FOCS 2010.
- [20] Kazuo Iwama, Takuya Nakashima, An Improved Exact Algorithm for Cubic Graph TSP, COCOON 2007.
- [21] David Eppstein, The Traveling Salesman Problem for Cubic Graphs, Journal of Graph Algorithms and Applications, 2007, 11(1) 61-81.
- [22] Shayan Oveis Gharan, Amin Saberi, Mohit Singh, A Randomized Rounding Approach to the Traveling Salesman Problem, FOCS 2011.
- [23] Tobias Mömke, Ola Svensson, Approximating Graphic TSP by Matchings, FOCS 2011.
- [24] S. Lin, B. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. Operations Research, 1973, 21(2), 498-516.
- [25] G. Reinelt. TSPLIB. Universität Heidelberg, Institüt für Informatik, Im Neuenheimer Feld 368,D-69120 Heidelberg, Germany, 2004. URL http://www.iwr.uniheidelberg.de/groups/comopt/software/TSPLIB95/
- [26] M. Jünger, G. Reinelt, G. Rinaldi, The travelling salesman problem, Handbooks in Operations Res. & Management Sc., Elsevier, 1995.
- [27] Gerhard Woeginger, Exact algorithms for NP-Hard problems, A survey, Combinatorial Optimization 2001, 185-208.
- [28] D. S. Johnson, L. A. McGeoch, E. E. Rothberg, Asymptotic experimental analysis for the Held-Karp traveling salesman

- bound, SODA, 1996.
- [29] NEOS Server for Optimization, http://neos-server.org/neos/
- [30] Marco Dorigo, Luca Maria Gambardella, Ant colonies for the traveling salesman problem, BioSystems 1997.
- [31] S. Kirkpatrick, C. D. Gelatt, Jr., M. P. Vecchi, Optimization by simulated annealing, Science, May 1983.
- [32] S. Ilker Birbil, Shu-Cherng Fang, Electromagnetism-like mechanism for global optimization, Journal of Global Optimization, 2003, 25, 263-282.
- [33] Robert Bosch, Opt Art, Math Horizons, February 2006, 14(3), 6–9.
- [34] Times-Ten Team: Mid-tier caching: the TimesTen approach, (Now Oracle cache and in memory database), SIGMOD 2002, 588–593.
- [35] SAP HANA, Realtime in memory technology, http://www.sap.com/hana/demos/index.epx

- [36] G. Graefe and W. J. McKenna, Extensibility and search efficiency in the Volcano optimizer generator, ICDE, 1993, 209–218.
- [37] P. G. Selinger, M.M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. Price, Access path selection in relational database management system, In ACM SIGMOD Intl. Conf. Management of Data, 1979, 23–34.
- [38] Foto N. Afrati, Jeffrey D. Ullman, Optimizing Multiway Joins in a Map-Reduce Environment, IEEE TKDE 2011, 23(9): 1282-1298
- [39] M. Hong, M. Riedewald, C. Koch, J. Gehrke, and A. Demers, Rule-Based Multi-Query Optimization, EDBT, 120–131, 2009
- [40] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis, An analysis of several heuristics for the traveling salesman problem, SICOMP 563–581, 1977.