OPTIMIZATION OF A 532-CITY SYMMETRIC TRAVELING SALESMAN PROBLEM BY BRANCH AND CUT

M. PADBERG *

Graduate School of Business Administration, New York University, Washington Square, New York, NY 10003, USA

G. RINALDI **

Istituto di Analisi dei Sistemi ed Informatica del CNR, viale Manzoni 30, 00185 Rome, Italy

Received July 1986
Revised January 1987

We report the solution to optimality of a 532-city symmetric traveling salesman problem involving the optimization over 141,246 zero–one variables. The results of an earlier study by Crowder and Padberg [1] are cross-validated. In this note we briefly outline the methodology, algorithms and software system that we developed to obtain these results.

traveling salesman problem * combinatorial optimization * polyhedral methods * scientific computation * software design * branch and cut

In Figure 1 we display an optimal solution to a symmetric traveling salesman problem (TSP) having 532 cities or 141,246 zero–one variables. At the time of writing this note (July, 1986) this is the largest TSP optimized to date. The previously largest TSP with a published optimal solution had 318 cities or 50,403 zero–one variables; see Crowder and Padberg [1]. The data for the 532-city problem are included in Table 1. The data are pairs of (pseudo-Euclidean) (x, y)-coordinates of 532 cities located in the continental United States in the order of the solution displayed in Figure 1. The intercity distances are calculated by the following FORTRAN procedure:

$$F1 = IX(I) - IX(J),$$
$$F2 = IY(I) - IY(J),$$
$$XDIST = (F1**2 + F2**2)/10.0,$$
$$XDIST = SQRT(XDIST),$$
$$NDIST = XDIST,$$
$$IF(NDIST \cdot LT \cdot XDIST)NDIST = NDIST + 1,$$

where $IX(I)$, $IY(I)$ are the coordinates of city I and $SQRT$ is the single-precision square-root function of the standard FORTRAN library. The coordinates and the formula for the calculation of the distance $NDIST$ between city I and city J account for the spherical curvature of the North American continent. The traveling salesman problem consists of finding the shortest roundtrip (tour, Hamiltonian cycle) passing through each city exactly once.

We are grateful to Shen Lin of AT&T Bell Laboratories for making the data available to us and thus challenging us to find a provably optimal solution to this large-scale problem. The algorithm that we developed to find the solution to this problem is based on the polyhedral theory for TSP's described in Grötschel and Padberg [5] and Padberg and Grötschel [9]. Its roots lie in the seminal work by Dantzig, Fulkerson and Johnson [3]; see also Crowder and Padberg [1], Grötschel [4] and Padberg and Hong [10]. The complete

* Partial financial support under NSF Grant no. DMS8508955 and NBS Contract no. 41-USC-252-C3.

** Partial financial support under a New York University Research Challenge Fund grant.

0167-6377/87/$3.50

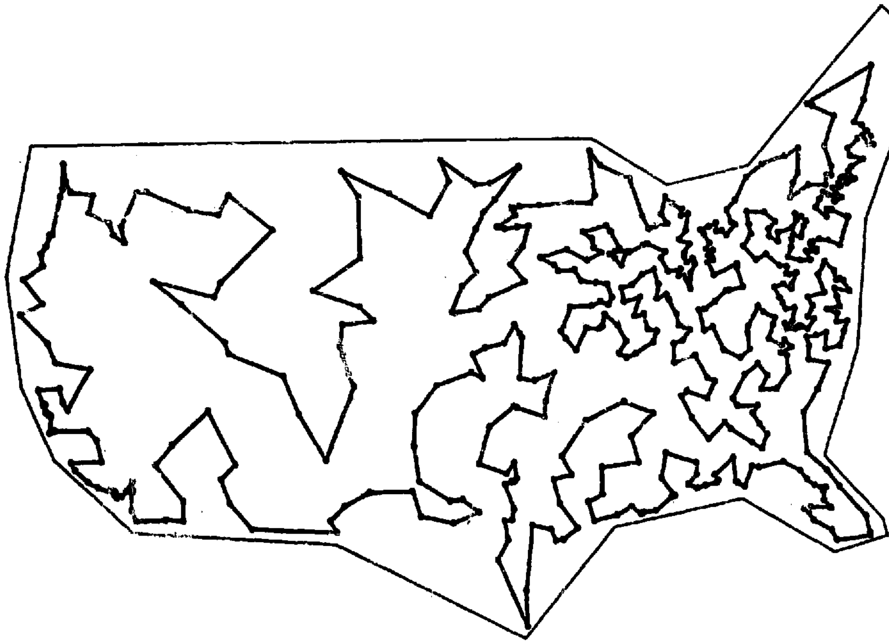

Fig. 1. Optimal Hamiltonian cycle on 532 nodes of length 27,686.

results of our current computational study will be reported in Padberg and Rinaldi [11].

To calculate the optimal tour displayed in Figure 1, we developed a software system written entirely in FORTRAN whose main characteristics and program flow are shown in Figure 2. The software system TSP has four major components:
- a heuristic procedure,
- a linear program solver,
- a constraint or cut generator,
- a branch and cut procedure.

The entire software system (not counting its numerous comment cards) currently has 9,701 lines of FORTRAN code, of which 827 lines account for the heuristic, 3,197 lines for the linear program solver, and 5,677 lines for both the constraint generator and the branch and cut procedure.

The current version of our program uses the XMP software package for linear programming written by Roy Marsten of the University of Arizona (see Marsten [8]), with a few minor changes. The heuristic is our adaptation of the exchange heuristic due to Lin and Kernighan [7], to accommodate symmetric TSP's of truly large size. For the 532-city problem this algorithm was executed 50 times, yielding 50 'local optima' whose objective function values range between 28,150 and 29,143. In view of the optimum tour length of 27,686 produced by our algorithm, the best heuristic tour thus deviates from optimality by a relative error of 1.7%. The heuristic is executed 50 times for two reasons: Firstly, we want the best possible upper bound that we can get within a reasonable amount of time. Secondly, the union of the edge-sets of the 50 locally optimal tours found by the heuristic defines a sparse partial graph, having in this case 1,278 edges on 532 nodes, that consists of 'reasonably selected' edges. This is desirable because the partial graph is used to initialize the LP-based procedure. The number 50 is, of course, perfectly arbitrary.

The LP-based constraint-generation procedure uses both column generation/column deletion and

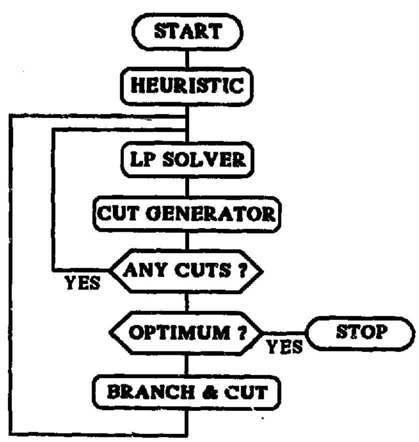

Fig. 2. Flowchart of software system TSP.

Table 1
(x, y)-coordinates of 532 cities, in the order of the solution displayed in Figure 1

3777	1322	4121	1334	4354	1388	4373	1311	4425	1258	4422	1249	4465	1205
4543	1170	4550	1219	4472	1284	4513	1330	4620	1408	4687	1373	4700	1242
4792	1342	4841	1360	4907	1286	4961	1355	4918	1349	4897	1388	4921	1416
4976	1432	4997	1406	5015	1430	5035	1478	5085	1434	5109	1380	5164	1440
5112	1459	5106	1515	5089	1562	5166	1585	5258	1612	5326	1485	5257	1501
5251	1458	5249	1453	5288	1300	5284	1284	5578	1315	5551	1434	5555	1519
5594	1578	5603	1598	5622	1583	5632	1590	5659	1531	5684	1528	5689	1591
5692	1626	5772	1570	5793	1666	5784	1788	5679	1777	5611	1783	5555	1772
5565	1700	5455	1689	5402	1674	5363	1733	5349	1794	5403	1768	5495	1799
5560	1925	5650	1916	5615	2004	5542	2021	5460	1972	5384	2095	5412	2264
5534	2221	5621	2185	5755	2241	5865	2095	5791	2008	5882	1861	5879	1787
5953	1781	5951	1744	5919	1683	6054	1709	6093	1703	6055	1790	6105	1870
6196	1801	6242	1762	6296	1724	6270	1640	6400	1638	6442	1657	6440	1710
6491	1680	6472	1599	6434	1550	6453	1478	6344	1436	6331	1499	6222	1465
6140	1529	6070	1382	5906	1472	5765	1427	5839	1327	5928	1288	5908	1260
5918	1223	6010	1144	6012	1291	6087	1253	6175	1210	6232	1329	6250	1226
6334	1213	6307	1119	6412	1131	6559	1143	6614	1271	6455	1294	6501	1385
6529	1475	6610	1437	6744	1417	6750	1223	7021	1281	7266	1379	7649	1276
7791	1052	7925	903	8166	607	8320	538	8351	527	8447	840	8359	904
8295	1094	8224	1159	8203	1206	8173	1147	8145	948	8084	1034	7995	1052
7954	1031	7970	1069	7954	1143	7909	1227	7838	1310	7782	1489	7877	1716
7878	1780	8091	1818	8068	1892	8057	1914	7830	1980	8025	2128	8073	2086
8097	2097	8147	2200	8190	2278	8167	2367	8056	2306	7944	2329	7899	2639
8152	2636	8296	2481	8317	2511	8483	2638	8516	2754	8528	2753	8532	2773
8476	2874	8587	2996	8679	3202	8505	3292	8409	3168	8148	3218	8035	2880
7798	2993	7535	2825	7471	3125	7721	3451	8111	3626	8272	3495	8348	3660
8575	3561	8777	3344	8916	3466	8938	3536	8832	3600	8827	3788	9475	3739
9820	3663	9225	4062	9005	3996	8832	4063	8812	3992	8706	3993	8479	4122
8440	4064	8436	4034	8400	4018	8253	4072	8372	4127	8326	4413	7947	4373
7707	4173	7715	4136	7752	3855	7600	3872	7310	3836	7421	4015	7424	4159
7027	4203	6990	4234	6913	4301	7110	4369	7187	4548	7275	4656	7489	4520
7452	4644	7640	4958	7944	5120	8266	5076	8598	4962	8737	4632	8698	4513
8797	4327	8944	4563	8934	4888	8705	5007	8787	5413	8967	5633	9132	5742
9231	5655	9345	6485	9135	6748	8917	6872	8746	6760	8272	7121	8665	7411
8888	7537	9194	7206	9385	7171	9401	7342	9468	7629	9371	7689	9202	7717
9172	7710	9080	7726	9193	7771	9250	7810	9271	7856	9250	7879	9230	7883
9213	7878	9197	7919	9227	7920	9205	8050	9171	8150	9073	8298	9005	8349
8947	8060	8746	8139	8669	8239	8722	8560	8753	8605	8483	8619	8576	8643
8574	8654	8556	8682	8486	8695	8492	8719	8354	8787	8304	8580	8435	8530
8499	8473	8064	8323	8057	8668	7880	8778	7645	9096	7503	8892	7293	9122
7128	8954	7016	8991	6929	8958	6809	8935	6799	8914	6415	8906	6336	8896
6087	8933	6252	8882	6354	8815	6349	8596	6533	8607	6595	8391	6707	8326
6765	8212	6611	8269	6247	8180	6228	8085	6336	7650	6336	7348	6120	7281
6391	6790	6998	7214	7146	7250	7096	7869	7275	7557	7576	7065	7680	7006
7804	6438	8149	6224	8549	5887	7787	5742	7679	5813	7501	5899	7419	5943
7203	5958	7112	5671	6997	5825	6918	6297	6518	5903	5922	6024	5699	6226
5840	5736	5992	5308	5615	5182	5420	5300	5606	4915	5568	4778	5352	4530
5362	4526	5721	4705	5864	4790	6042	4839	6279	4900	6468	4768	6624	4880
6901	4936	6823	4674	6687	4595	6471	4275	6328	4438	6136	4352	5916	4326
5953	4438	5987	4558	5986	4636	5838	4477	5781	4525	5776	4498	5746	4453
5698	4261	5622	3964	5589	3776	5512	3747	5079	3873	5207	3745	5284	3447
5628	3261	5749	3177	5713	3124	5663	3009	5584	3081	5404	3074	5461	2993
5498	2895	5562	2891	5536	2828	5704	2820	5800	2873	5828	2766	5921	2799
5960	2792	6011	2756	5963	2707	6009	2678	6113	2705	6174	2706	6263	2679
6151	2767	6157	2815	6085	2805	6130	2925	6009	3012	5942	2982	5895	3168
5918	3206	6006	3329	6017	3354	6149	3381	6088	3454	6062	3511	6023	3461
6007	3412	5986	3426	5837	3535	5788	3589	5887	3796	5936	3705	6022	3675
6061	3591	6157	3715	6088	3925	6208	4167	6261	4021	6313	3972	6273	3817

Table 1 (continued)

6276	3816	6369	3732	6362	3592	6358	3483	6478	3413	6561	3435	6539	3513
6595	3593	6642	3790	6901	3841	6963	3782	7056	3662	6819	3601	6807	3482
6781	3455	6659	3453	6744	3311	6882	3202	7013	3251	7041	3169	6982	3088
6845	2942	6731	2928	6729	3019	6588	3082	6623	3172	6428	3145	6502	3291
6371	3336	6322	3245	6206	3167	6135	3063	6272	2992	6417	2984	6359	2897
6404	2867	6529	2772	6640	2730	6665	2614	6780	2664	6822	2745	7010	2710
6902	2515	7098	2366	7118	2284	7298	2332	7267	2535	7324	2585	7518	2446
7692	2247	7497	2117	7556	2045	7649	1817	7691	1714	7539	1684	7454	1781
7364	1865	7280	1828	7320	2028	7260	2083	7007	2007	6873	1894	6972	1786
7113	1823	7089	1674	6901	1589	6744	1629	6730	1692	6655	1600	6601	1679
6657	1698	6611	1833	6470	1816	6401	1811	6361	1933	6316	1991	6528	2056
6595	2050	6749	2001	6849	2077	6801	2251	6596	2401	6649	2476	6459	2562
6441	2509	6342	2419	6220	2334	6212	2299	6152	2174	5976	2268	5938	2270
6088	2480	5972	2555	5939	2511	5910	2530	5908	2565	5848	2529	5848	2595
5783	2575	5702	2644	5670	2682	5623	2608	5574	2543	5657	2525	5637	2472
5676	2419	5557	2353	5548	2392	5429	2462	5321	2397	5075	2326	4913	2195
4798	1990	4701	1878	4943	1837	5051	1865	5200	1873	5042	1715	5017	1549
4915	1556	4913	1510	4821	1526	4639	1629	4327	1585	4270	1808	4255	1868

row generation/row deletion. To optimize the 532-city problem a total of 800 calls to the XMP subroutines PRIMAL (for the primal simplex method) or DUAL were executed. The initial LP has 532 rows and 1,278 columns. The subsequent LP's never had more than 815 rows and 1,520 columns, not counting the slack variables. Thus the bulk of the 141,246 structural variables over which we have to optimize in order to solve the 532-city problem are never included into the 'active' set of columns for the LP calculations. They are 'priced' out in a subroutine and 'forgotten' as soon as this becomes mathematically possible using the simplex reduced cost, the upper bound and the LP lower bound on the optimal objective function value. The row generation/row deletion consists of generating linear inequalities that define facets of the convex hull of tours that are violated by the optimal solution to the current LP relaxation of the problem. These inequalities are added to the existing linear programming problem, but are dropped again (and put into a memory bank) as soon as their corresponding slacks become basic after reoptimization. Also, in order to keep the basis of the LP 'clean', we add only a selected few out of the total set of constraints generated in a given round for the actual reoptimization. The other constraints are 'suspended' temporarily and 'resumed' in the next iteration if they remain violated by the new LP optimum. The constraints that we generate are subtour elimination constraints (SEC's) and comb constraints (combs); see Grötschel and Padberg [5].

The SEC identification is carried out *exactly* at every step of the procedure by a very effective, new *minimum capacity cut* algorithm that is described in Rinaldi and Padberg [12]. The comb identification is, from a theoretical point of view, incomplete at present because there is no exact algorithm known to solve the associated separation problem. Nevertheless, our heuristic procedures – geared to identify general combs – have proven to be very effective. They will be decribed in the complete report.

To evaluate our constraint generator we executed our software system on Euclidean test problems of the Crowder and Padberg study. In the five 100-city problems as well as the 318-city Hamiltonian path problem, the program found the optimal tours by linear programming and constraint generation only, which is a good *certificate of optimality* of an incidence vector of a tour. In other words, different form the earlier study no recourse to an enumerative or branch and bound technique was necessary to establish (and thereby cross-validate!) the results of that study. To test the constraint generator further, we solved the 318-city problem also as a *Hamiltonian cycle* problem by dropping the requirement that a particular start and end point had to be used. Our program found the optimal solution to this *new* problem to have an objective function value of 42,029 miliinches. It is thus 3,185 milliinches shorter than the optimal solution to the *Hamilto-*

nian path problem using the prescribed start and end point published by Crowder and Padberg [1] which has 41,345 milliinches while the 'enforced' arc is 3,869 milliinches long.

The linear description of the underlying TSP polytope that we work with is incomplete as of today since SEC's and combs are not sufficient to describe the convex hull of tours, and our comb generator is at present not exact in a theoretical sense. It must therefore be expected that the procedure will come to a point where the current LP optimum solution cannot be cut off by the constraints generated by our software system. Also, if we detect a 'long' sequence of small gains in the objective function, i.e., if we detect a 'tailing off' phenomenon, we might just want to stop the cut generation and 'try something else' in order to speed the convergence of the calculations. If the solution vector does not correspond to a tour at such a point, we must then resort to some enumerative technique such as branch and bound. We initially used a standard branch and bound procedure (written by Roy Marsten). Its performance (as would have been the case with any other branch and bound code on our problem) turned out to be quite unsatisfactory. For example, on the 48-city problem described in Rinaldi and Yarrow [13], the branch and bound code required *40 minutes* of CPU time on a VAX 11/780 to find the optimal tour, when the cut generator was restricted to SEC's only.

We therefore developed a *'branch and cut'* scheme where, like in branch and bound, we select a variable x, say, by some reasonable criterion to branch on. Enforcing the constraint $x \geq 1$ on the 'up-branch', $x \leq 0$ on the 'down-branch', we move from the current vertex to a different one which is fed into the cut generator which then (typically) finds new constraints to cut off the new vertex, etc. When this procedure was applied iteratively ceteris paribus to the aforementioned 48-city problem, the total execution time was reduced to a mere *25 seconds* on the same machine. The constraints that the cut generator generates at any given node of the search tree are, of course, valid at *any other* node of the tree because they define facets of the entire TSP polytope. This means that it is no longer desirable to keep track of the order in which the constraints are generated which would be necessary if a standard branch and bound code were used. As a consequence, the amount of book-keeping can be kept to the minimum necessary to represent the search tree. This also means thart there is no need to keep track of the information on previous LP bases and the like, all of which accounts for the lion's share of the storage requirements of a typical branch and bound code. In the context of branch and cut it makes perfectly sense to forget this information as at every branching step we do not just impose simple constraints of the type $x \geq 1$ or $x \leq 0$, but rather generate any number of violated facet-defining constraints repeatedly as long as the progress towards optimality remains satisfactory.

In Figure 3 we display the branch and cut tree that was produced for the 532-city problem. The top node of the tree is the optimal objective function value of the linear program having 532 equations (expressing the conditions that every node must be met by exactly two edges of a tour) and the 141,246 structural variables of the overall problem. The node below it is the optimal objective function value after the first run through the cut generator over an enlarged set of constraints, but the same set of 141,246 structural variables. Then branching takes place and two problems are created that are optimized sequentially over the entire column set but with new constraints generated (some old constraints dropped) in the course of calculations. For reasons of space, we print only the two or three digits before and after the decimal point, the objective function values being always greater than 27,600. We used a VAX 11/780 and a Micro VAX II computer to calculate the optimal tour. The run that produced the tree of Figure 3 took about 60 hours of CPU time, of which the heuristic consumed about 4 hours. For the final runs we intend to work on a faster computer and details on computation times will be contained in the complete report.

We have experimented both with a depth-first and breadth-first search strategy, but decided to adopt the breadth-first strategy (which generated the tree of Figure 3). The reason for choosing breadth-first is that the heuristic, for large-scale problems, fails to find the optimum. If the gap between the best heuristic tour and the optimum tour is 'reasonably' large, then a depth-first strategy, typically, produces unnecessarily large search trees. As we expected the number of nodes of the breadth-first branch and cut tree has very few nodes indeed, e.g. as compared to the branch-

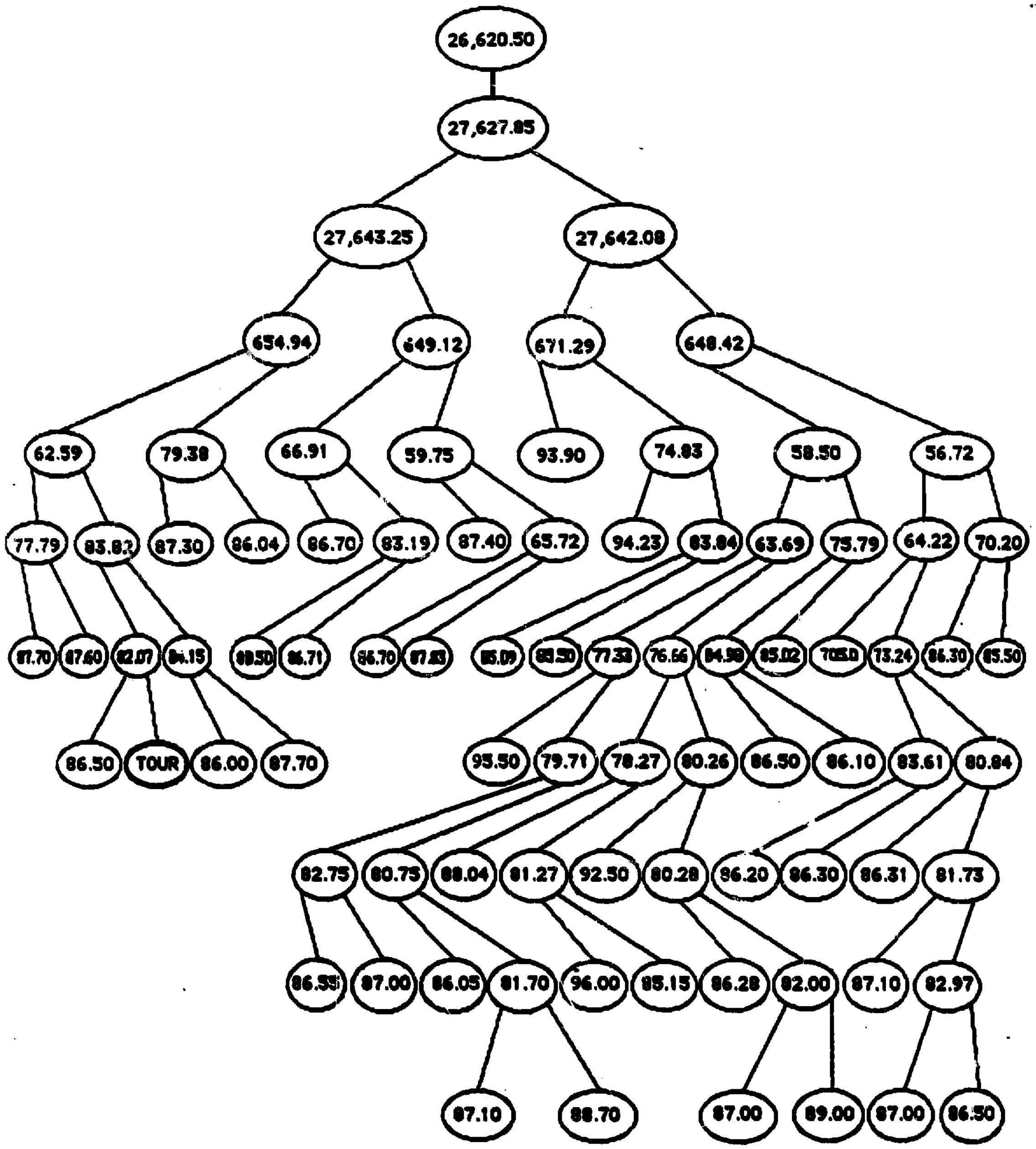

Fig. 3. Branch and cut tree for the 532-city problem.

ing trees published by Held and Karp [6], for TSP's of size less than 65 cities. Branch and cut is, of course, applicable to any combinatorial optimization problem provided that one works with the 'global' cuts that are derived from the polyhedron associated with the overall problem. Much work remains to be done along the lines of this new idea of marrying cutting planes with branching techniques in integer and mixed-integer programming.

Note added (December, 1986): In the interim, we have had access to the CYBER 205 Supercomputer of the National Bureau of Standards in Gaithersburg, MA and have implemented an executable version of our software package for CDC computers. The 532-city problem was rerun and terminated with the same optimal tour after 5 hours and 58 minutes of computing time on the CYBER 205.

Since then we have made several improvements to the original code. Among these there are the generation of clique-tree inequalities (see Grötschel and Padberg [5]), and several means to accelerate the search for an optimal solution in a branching scheme that employs a breadth-first search. With these improvements, we have been able to solve to optimality (Euclidean) real-world problems having 1,002 and 2.392 cities, respectively. The CPU times on the NBS CYBER 205 for the 1,002-city problem were 7 hours and 18 minutes and 27 hours and 20 minutes for the 2,392-city problem, respectively.

We are most grateful for the generous support and assistance given to us by our colleagues of the National Bureau of Standards.

The details of these new developments will be included into the complete report as well.

Acknowledgments

We gratefully acknowledge the generous and cooperative support given to us during the course of this study from the GBA Academic Computing Center at New York University and the Centro di Calcolo at IASI-CNR in Rome.

References

[1] H. Crowder and M. Padberg, "Solving large-scale symmetric traveling salesman problems to optimality", *Management Science* **26**, 495–509 (1980).

[2] G. Dantzig, R. Fulkerson and S. Johnson, "Solution of a large-scale traveling salesman problem", *Operations Research* **2**, 393–410 (1954).

[3] G. Dantzig, R. Fulkerson and S. Johnson, "On a linear programming combinatorial approach to the traveling salesman problem", *Operations Research* **7**, 58–66 (1959).

[4] M. Grötschel, On the symmetric travelling salesman problem: Solution of 120-city problem", *Mathematical Programming Studies* **12**, 61–77 (1980).

[5] M. Grötschel and M. Padberg, "Polyhedral theory", in: E. Lawler et al., eds., *The Traveling Salesman Problem*, Wiley, Chichester, 1985, 251–305.

[6] M. Held and R. Karp, "The traveling salesman problem and minimum spanning trees: Part II", *Mathematical Programming* **1**, 6–26 (1971).

[7] S. Lin and B. Kernighan, "An effective heuristic algorithm for the traveling salesman problem", *Operations Research* **21**, 498–516 (1973).

[8] R. Marsten, "The design of the XMP linear programming libary", *ACM Transactions of Mathematical Software* **7**, 481–497 (1981).

[9] M. Padberg and M. Grötschel, "Polyhedral computations", in: E. Lawler et al., eds., *The Traveling Salesman Problem*, Wiley, Chichester, 1985, 307–360.

[10] M. Padberg and S. Hong, "On the symmetric traveling salesman problem: A computational study", *Mathematical Programming Studies* **12**, 78–107 (1980).

[11] M. Padberg and G. Rinaldi, "An LP-based algorithm for the resolution of large-scale traveling salesman problems", Preprint, New York University, New York, in preparation.

[12] G. Rinaldi and M. Padberg, "An efficient algorithm for the minimum capacity cut problem in large sparse graphs", Preprint, IASI-CNR, Rome, 1985.

[13] G. Rinaldi and L.A. Yarrow, "Optimizing a 48-city traveling salesman problem: A case study in combinatorial problem solving", Preprint R.122, IASI-CNR, Rome, 1985.