

# Application of column splitting to the travelling salesman problem

John F. Raffensperger, PhD

Dept. of Management, Private Bag 4800

University of Canterbury, Christchurch, New Zealand

j.raffensperger@mang.canterbury.ac.nz

29 October 2004

---

## Abstract

In this paper, we apply column splitting to the Travelling Salesman Problem (TSP), producing two different decompositions. The lower bounds on these models are at least as good as the standard Held-Karp lower bound, and we give examples with strictly better lower bounds. The bounds appear to be equivalent to the standard Dantzig, Fulkerson & Johnson formulation, with all subtour breaking and 2-matching constraints added.

---

## 1 The DFJ formulation, the 1-tree and the 2-matching.

The travelling salesman problem (TSP) is often modelled with the well-known Dantzig, Fulkerson & Johnson (1954) model:

Indices:  $i, j$  city,

Parameters:  $n$  = number of cities;  $c_{ij}$  = cost to travel from city  $i$  to city  $j$ .

Decision variables:  $x_{ij} = 1$  if the salesman should travel between cities  $i$  and  $j$ , else 0.

1. Model DFJ:  $\min \sum_i \sum_{j>i} c_{ij} x_{ij}$ ,
2.  $\sum_{j>i} (x_{ij} + x_{ji}) = 2$  for all  $i$ .
3.  $\sum_{i,j \in S} x_{ij} \leq |S| - 1$ , for every subset  $S$  such that  $3 \leq |S| \leq \lfloor n/2 \rfloor$ , and  $\{1\} \notin S$ .
4.  $x_{ij} \in \{0, 1\}$  for all  $i, j: j>i$ .

Explanation:

1. Minimise total distance travelled.
2. Each city must have degree 2, i.e., must be entered and departed.
3. No subset of cities  $S$  consist of a tour, unless that subset as all the cities.
4. The salesman cannot split his travel between cities.

We will denote the LP relaxation of DFJ, 1 through 3, as  $v(\underline{\text{DFJ}})$ .

In this paper, we apply column splitting to the TSP, and show that column splitting provides a bound tighter than  $v(\underline{\text{DFJ}})$ .

### 1.1 The 1-tree relaxation and the Held-Karp lower bound

Held & Karp (1969) showed how to find  $v(\underline{\text{DFJ}})$ , with the following model:

5. Model HK:  $\min \sum_t \pi_t \sum_i \sum_{j>i} c_{ij} x_{ij}^t$ , where  $x_{ij}^t$  is the set of arcs in 1-tree  $t$ ,
6.  $\sum_i \pi_t (\sum_{j>i} (x_{j,i}^t + x_{i,j}^t)) = 2$ , for all  $i$  (dual prices  $\lambda_i$ ),
7.  $\sum_t \pi_t = 1$ ,
8.  $\pi_t \geq 0$ , for all  $t$ .

A solution method for Model HK is as follows:

Step 0. Pick dual prices  $\lambda_i$ .

Step 1. Solve the 1-tree:  $\min \sum_i \sum_{j>i} (c_{ij} - \lambda_i - \lambda_j) x_{ij} + 2 \sum_i \lambda_i$ ,  $x \in T$ .

Step 2. Add one variable  $\pi_{t+1}$  to Model HK above and resolve. Update dual prices  $\lambda_i$  from constraint set 6. Quit when  $v(\text{HK}^t) = v(1\text{-tree}^t)$ .

This requires many iterations, with a “tail-off” effect. To improve on this method, Held & Karp (1970) gave a subgradient optimisation algorithm to find  $v(\text{HK})$ , and sometimes an optimal solution to the TSP. The algorithm optimises over  $\lambda_i$ , with a price for each node. As we shall see, we can do better if we switch to optimisation arc-wise, with a price for each arc.

## 1.2 The 2-matching relaxation

Instead of relaxing the degree 2 constraints, set 2, we could relax the subtour breaking constraints, set 3, resulting in a 2-matching problem. The 2-matching problem can be solved in polynomial time, but the LP formulation with constraints 1 and 2 is not naturally integer. However, Edmonds (1965) showed that constraints 1, 2 and 9 below give a complete linear description of the 2-matching problem.

9.  $\sum_{(i,j):i \in W, j \in W} x_{ij} + \sum_{(i,j) \in F} x_{ij} \leq |W| + \lfloor |F|/2 \rfloor$ , for all node subsets  $W$ , and all sets of arcs  $F$ , where  $(i,j) \in F$  if  $i \in W$  xor  $j \in W$ .

These constraints 9 are called the *2-matching inequalities*, not to be confused with the constraint set 2 that requires that nodes have degree 2. The 2-matching inequalities are part of a large set of constraints called comb constraints; set  $W$  is often called the “handle”, and the set  $F$  is called the “teeth”. Unfortunately, the 2-matching problem has an exponential number of two matching inequalities.

Nemhauser & Wolsey (1988) give the following model:

10. Model NW:  $\min \sum_{t=1}^T \alpha_t (\sum_{i,j} c_{i,j} x_{i,j}^t)$

11.  $\sum_{t=1}^T \alpha_t (\sum_{i,j} x_{i,j}^t) \leq |S| - 1$ , for every subset  $S$  such that  $3 \leq |S| \leq \lfloor n/2 \rfloor$  and  $\{1\} \notin S$ .

12.  $\sum_{t=1}^T \alpha_t = 1$

13.  $\alpha_t \geq 0$  for all  $t = 1, \dots, T$ .

Model NW finds a convex combination of 2-matchings that satisfies the subtour breaking constraints. Nemhauser & Wolsey reference Balas & Christofides (1983) for this model, but the Balas & Christofides article never shows the above master. Balas & Christofides instead write the subtour breaking constraints in generic form of  $\sum_{i,j} a_{i,j}^s x_{i,j} \geq a^s_0$  for  $s \in S$ , and give the following relaxation:

14. Model BC:  $\min \sum_{i,j} c_{i,j} x_{i,j} - \sum_s w_s (\sum_{i,j} a_{i,j}^s x_{i,j} - a^s_0)$

subject to 2 and 4. If we wish, we can drop 4 and add 9.

Using Model BC, Balas & Christofides give an algorithm to solve the TSP (actually using the assignment problem for asymmetric TSPs rather than the 2-matching).

Nemhauser & Wolsey note that  $v(\text{DFJ}) \leq v(\text{NW}) = v(\text{BC})$ , because the latter two formulations satisfy the 2-matching inequalities 9. Column splitting appears to give a result at least as good as the strongest of these, but without the need to relax a potentially exponential number of subtour breaking rows.

## 2 Application of column splitting to the TSP

Guignard and Kim (1987) showed that Lagrangean decomposition based on column splitting dominates ordinary Lagrangean decomposition. To apply column splitting to Model DFJ, we must add a decision variable  $w_{ij}$ , which is a clone of  $x_{ij}$ . Model DFJ is repeated here with the new variables.

15. Model DFJ1:  $\min \sum_i \sum_{j:j>i} c_{ij} x_{ij}$ ,

16.  $\sum_{j:j>i} (w_{ij} + w_{ji}) = 2$  for all  $i$ ,

17.  $\sum_{i,j \in S} x_{ij} \leq |S| - 1$ , for every subset  $S$  such that  $3 \leq |S| \leq \lfloor n/2 \rfloor$  and  $\{1\} \notin S$ ,

18.  $w_{ij} = x_{ij}$  for all  $i, j:j>i$ ,

19.  $w_{ij} \in \{0, 1\}$  for all  $i, j:j>i$ .

If we relax constraint set 18 into the objective, we obtain Model CS-DFJ.

Model CS-DFJ:  $\min \sum_i \sum_{j>i} c_{ij} x_{ij} - \sum_i \sum_{j>i} \beta_{ij} (x_{ij} - w_{ij})$ ,

subject to 16, 17, and 19. Because constraint 18 is an equality,  $\beta_{ij}$  is unrestricted in sign.

Model CS-DFJ decomposes into a 1-tree and a 2-matching, both of which can be solved in polynomial time.

For convenience, partly to avoid row generation of constraint set 17 in our test algorithm, we chose to explore alternate formulations to DFJ.

### 3 Column splitting into $n-1$ max flow problems and a 2-matching problem

A lower bound for the objective value of DFJ is the objective value of its LP relaxation, which we denote by underscore, DFJ. DFJ is typically solved by adding subsets of subtour breaking cuts to DFJ as needed, as with Dantzig, Fulkerson & Johnson (1954) and Applegate, Bixby, Chvatal, and Cook (1998). The subtour breaking cuts may be found with a maximum flow model.

Martin (1991) observed that the matrix from the maximum flow model could be applied to write a totally dual integral LP for the minimum spanning tree, as follows.

Indices:  $i, j, k =$  nodes.

Variables:  $x_{i,j} = 1$  if arc  $i, j$  is in the tree, else 0, for all  $i$ , for all  $j > i$ .

$z_{i,j,t}$  = flow to node  $t$  across arc  $(i,j)$ , for all  $i:t>i, j:j>i$ , and all  $t$ . Let  $z_{i,j,i} = 0$ .

20. Model MST-LP:  $\min \sum_i \sum_{j>i} c_{ij} x_{ij}$ ,

21.  $\sum_i \sum_{j>i} x_{i,j} = n - 1$ ,

22.  $x_{i,j} = z_{i,j,t} + z_{j,i,t}$ , for all  $i$ , for all  $j:j > i$ , and all  $t$ .

23.  $\sum_{j \neq t} (z_{i,j,t} + z_{j,i,t}) \leq 1$ , for all  $i:t>i$ , and all  $t$ ,

24.  $x_{i,j}, z_{i,j,t} \geq 0$ .

MST-LP can be modified for the TSP, in two different ways. Mainly, we must constrain the degree of 2 at each node. We show the first way in this section and the second way in the next section.

In the first way, the interpretation of the model follows from the maximum flow model. In the  $z_{ijt}$  variables, the flow into each node  $t$  must be 2, and the flow in equals flow out for all other nodes  $i, j \neq 1$ . As with the maximum flow model, node 1 is special as it serves as the source.

25. Model TSP-mflow:  $\min \sum_i \sum_{j>i} c_{ij} x_{ij}$ ,

26.  $x_{i,j} = z_{i,j,t} + z_{j,i,t}$ , for all  $i$ , for all  $j:j > i$ , and all  $t$ ,

27.  $\sum_{j>i} (x_{ij} + x_{ji}) = 2$  for all  $i$ .

28.  $\sum_i z_{i,j,t} = \sum_k z_{j,k,t}$  for all  $j:j \neq t$  and  $j \neq 1$ , for all  $t:t \neq 1$ ,

29.  $\sum_{i:i \neq t} z_{t,i,t} = 2$ , for all  $t \neq 1$ ,

30.  $z_{i,j,t} \geq 0$ ,

31.  $x_{ij} \leq 1$  for all  $i, j$ .

32.  $x_{ij} \in \{0, 1\}$  for all  $i, j$ .

Constraint set 27 may appear redundant to 29, but this is not the case. A solution in which a node has degree 4 satisfies the other constraints.

We apply column splitting to Model TSP-mflow by relaxing constraint set 26, the constraints that tie  $x_{ij}$  and  $z_{ijt}$ , to obtain the following.

33. Model CS-mflow:  $\min \sum_i \sum_{j>i} c_{ij} x_{ij} + \sum_i \sum_{j>i} \sum_t \theta_{i,j,t} (z_{i,j,t} + z_{j,i,t} - x_{i,j})$

$= \min \sum_i \sum_{j>i} (c_{ij} - \sum_t \theta_{i,j,t}) x_{ij} + \min \sum_i \sum_{j>i} \sum_t \theta_{i,j,t} (z_{i,j,t} + z_{j,i,t})$ ,

subject to 28 through 32.

With the first term in the minimisation, we have a 2-matching  $2M(\theta)$  with constraints 27 and 32. With the second term in the minimisation, we have  $n - 1$  maximum flow problems (with constraints 28, 29, and 30).

For the moment, we will drop constraint 27. This converts the 2-matching into a trivial variable-fixing problem  $VF(\theta)$ :

set  $x_{ij}$  to 1 if  $(c_{ij} - \sum_t \theta_{i,j,t}) < 0$ , or

set  $x_{ij}$  to 0 if  $(c_{ij} - \sum_t \theta_{i,j,t}) \geq 0$ .

Hence, at the optimum,  $\sum_i \sum_{j>i} x_{ij} (c_{ij} - \sum_t \theta_{i,j,t}) \leq 0$ , for all  $\theta_{i,j,t}$  and  $c_{ij}$ .

We next prove that this decomposition dominates HK.

For fixed  $\underline{\theta}$ , denote  $D(\underline{\theta}) = \min \sum_i \sum_{j>i} (c_{ij} - \sum_t \underline{\theta}_{i,j,t}) x_{ij} + \min \sum_i \sum_{j>i} \sum_t \underline{\theta}_{i,j,t} (z_{i,j,t} + z_{j,i,t})$ .

**Theorem:**  $v(\text{CS-mflow}) \geq v(\text{HK})$ .

Proof: Choose  $(\lambda^*, x^*, z^*)$  as the optimal solution to  $H(\lambda)$ . Let  $\theta_{i,j,t}^\circ \equiv (\lambda_i^* + \lambda_j^*)/n$  for all  $i, j, t$ . Since any  $x_{ij}$  is feasible to the variable fixing subproblem  $VF(\theta^\circ)$ , we can use  $x_{ij}^*$ . Choose  $z_{i,j,t}^\circ$  optimal to  $D(\theta^\circ)$ .

$$\begin{aligned}
34. D(\theta^\circ) - H(\lambda^*) &= \sum_i \sum_{j>i} (c_{ij} - \sum_t \theta_{i,j,t}^\circ) x_{ij}^* + \sum_i \sum_{j>i} \sum_t \theta_{i,j,t}^\circ (z_{i,j,t}^\circ + z_{j,i,t}^\circ) - \sum_i \sum_{j>i} (c_{ij} - \lambda_i^* - \lambda_j^*) x_{ij}^* - 2 \sum_i \lambda_i^* \\
&= \sum_i \lambda_i^* \sum_{j>i} \sum_t (z_{i,j,t}^\circ + z_{j,i,t}^\circ) - 2 \sum_i \lambda_i^* \\
&= \sum_i \lambda_i^* (\sum_{j>i} \sum_t (z_{i,j,t}^\circ + z_{j,i,t}^\circ) - 2) \\
&= 0, \text{ because } z_{i,j,t}^\circ \text{ is feasible to constraints 28 and 29.}
\end{aligned}$$

Now  $H(\lambda^*)$  has been maximised, but we may still be able to improve  $D(\theta)$  by choosing a better  $\theta$  and a better  $x_{ij}$  for the subproblem  $VF(\theta)$ . Thus, the result holds. When we add the 2-matching constraints 27, the value of  $D(\theta^\circ)$  must be even higher.

This decomposition requires  $O(n^3)$  variables, so it will be computationally demanding. However, our next decomposition requires only  $O(n^2)$  variables.

## 4 Column splitting into a 2-matching and a 1-tree

We can write a second TSP formulation based on MST-LP. We modify the data by cloning node 1 to node  $n+1$ . If we wish, we can set  $c_{1,n+1} = \infty$ . The interpretation of Model TSP-1tree below is a 1-tree with degree 2 constraints, so the optimal value of the LP relaxation to TSP-1tree must equal the optimal value of the LP relaxation of DFJ. We note the similarity of this model to that of Wong (1980) and Claus (1984).

35. Model TSP-1tree:  $\min \sum_i \sum_{j>i} c_{ij} x_{ij}$ ,
36.  $\sum_j (x_{ij} + x_{ji}) = 2$  for all  $2 \leq i \leq n$ ,
37.  $x_{i,j} = z_{i,j,t} + z_{j,i,t}$ , for all  $i$ , for all  $j: j > i$ , and all  $t$ ,
38.  $\sum_i \sum_{j>i} x_{ij} = n$ , (noting that we have  $n+1$  nodes),
39.  $x_{1,i} + x_{i,n+1} \leq 1$  for all  $2 \leq i \leq n$ ,
40.  $\sum_j (z_{i,j,t} + z_{j,i,t}) \leq 1$ , for all  $i: i > t$ , and all  $t$ ,
41.  $z_{i,j,t} \geq 0$ , for all  $i, j, t$ .
42.  $x_{ij} \in \{0, 1\}$  for all  $i, j: j > i$ .

Constraint set 38 is redundant to 36. If we drop constraint set 36, we have an LP formulation for the 1-tree. However, we use this constraint later.

We apply column splitting to Model TSP-1tree. To use column splitting, we introduce new variables  $w_{ij}$  into Model TSP-1tree, repeated here with the new variables.

43. Model TSP-1tree:  $\min \sum_i \sum_{j>i} c_{ij} x_{ij}$ ,
44.  $x_{i,j} = z_{i,j,t} + z_{j,i,t}$ , for all  $i$ , for all  $j: j > i$ , and all  $t$ ,
45.  $\sum_i \sum_{j>i} x_{ij} = n$ , (noting that we have  $n+1$  nodes),
46.  $x_{1,i} + x_{i,n+1} \leq 1$  for all  $2 \leq i \leq n$ ,
47.  $\sum_j (z_{i,j,t} + z_{j,i,t}) \leq 1$ , for all  $i: i > t$ , and all  $t$ ,

48.  $z_{i,j,t} \geq 0$ , for all  $i, j, t$ .  
 49.  $x_{ij} \in \{0, 1\}$  for all  $i, j: j > i$ .  
 50.  $w_{ij} = x_{ij}$  for all  $i, j: j > i$ .  
 51.  $\sum_j (w_{ij} + w_{ji}) = 2$  for all  $2 \leq i \leq n$ ,  
 52.  $w_{ij} \in \{0, 1\}$  for all  $i, j: j > i$ .

Relaxing constraint set 50 gives Model CS-1tree.

53 Model CS-1tree:  $\min \sum_i \sum_{j>i} c_{ij} x_{ij} - \sum_i \sum_{j>i} \beta_{ij} (x_{ij} - w_{ij})$ , subject to 45 through 49, and 51 and 52. Because constraint 44 is an equality,  $\beta_{ij}$  is unrestricted in sign.

We prove that  $v(\text{CS-1tree}) \geq v(\text{HK})$ .

For fixed  $\underline{\lambda}$ , denote  $H(\underline{\lambda}) = \min \sum_i \sum_{j>i} (c_{ij} - \underline{\lambda}_i - \underline{\lambda}_j) x_{ij} + 2 \sum_i \underline{\lambda}_i$ .

For fixed  $\underline{\beta}$ , denote  $C(\underline{\beta}) = \min \sum_i \sum_{j>i} c_{ij} x_{ij} - \sum_i \sum_{j>i} \underline{\beta}_{ij} (x_{ij} - w_{ij})$ .

**Theorem:  $v(\text{CS-1tree}) \geq v(\text{HK})$ .**

Proof: Choose  $(\lambda^*, x^*, z^*)$  as the optimal solution to  $H(\lambda)$ . Let  $\beta_{i,j}^\circ \equiv \lambda_i^*/n$  for all  $i, j$ . Therefore,  $\sum_{j>i} \beta_{i,j}^\circ = \lambda_i^*$ . Choose  $w_{ij}^\circ$  optimal to the 2-matching subproblem.

Then  $v(C(\beta^\circ)) - v(H(\lambda^*))$

$$\begin{aligned} &= \sum_i \sum_{j>i} c_{ij} x_{ij}^* - \sum_i \sum_{j>i} \beta_{ij}^\circ (x_{ij}^* - w_{ij}^\circ) - \sum_i \sum_{j>i} (c_{ij} - \lambda_i^* - \lambda_j^*) x_{ij}^* - 2 \sum_i \lambda_i^*, \\ &= \sum_i \sum_{j>i} c_{ij} x_{ij}^* - \sum_i \lambda_i^* \sum_{j>i} (x_{ij}^* - w_{ij}^\circ) - \sum_i \sum_{j>i} (c_{ij} - \lambda_i^* - \lambda_j^*) x_{ij}^* - 2 \sum_i \lambda_i^*, \text{ since } \sum_{j>i} \beta_{i,j}^\circ = \lambda_i^*, \\ &= \sum_i \sum_{j>i} c_{ij} x_{ij}^* - \sum_i \lambda_i^* \sum_{j>i} x_{ij}^* + \sum_i \lambda_i^* \sum_{j>i} w_{ij}^\circ - \sum_i \sum_{j>i} c_{ij} x_{ij}^* + \sum_i \lambda_i^* \sum_{j>i} x_{ij}^* - 2 \sum_i \lambda_i^*, \end{aligned}$$

rearranged,

$$\begin{aligned} &= \sum_i \lambda_i^* \sum_{j>i} w_{ij}^\circ - 2 \sum_i \lambda_i^*, \text{ rearranged,} \\ &= \sum_i \lambda_i^* (\sum_{j>i} (w_{ij}^\circ + w_{ji}^\circ) - 2), \text{ rearranged,} \\ &= 0, \text{ since } \sum_j (w_{ij}^\circ + w_{ji}^\circ) = 2. \end{aligned}$$

Now observe that  $x_{ij}^*$  is optimal to  $H(\lambda)$ , but CS-1tree has not been maximised over  $\beta_{i,j}$ . We therefore could increase the optimal value of CS-1tree, so  $v(\text{CS-1tree}) \geq v(\text{HK})$ .

## 5 Numerical examples

Example 1. We wanted to find a 10-city TSP with a relatively large gap between the optimal tour length and  $v(\text{DFJ})$ . We used a script in AMPL (Fourer, Gay & Kernighan 2003) to generate random 10-city TSPs, and then to solve them to calculate the duality gap. We first solved the LP relaxation of TSP-1tree, then the IP. Eventually, we selected the problem with the following coordinates.

$x$  coordinates: 0      10      35      40      54      56      65      66      74      95,  
 $y$  coordinates: 0      69      97      51      58      28      80      86      30      45.

The optimal tour length is 346.7837104, and  $v(\text{DFJ})$  is 340.3072057.

We solved CS-1tree with the following Model CS, obtaining prices  $\beta_{ij}$  from constraint set 57. We note that this formulation is quite general. The subproblems are the 1-tree and the 2-matching problems. Convergence was slow, but obtained the optimal value of 346.7837104.

54. Model CS:  $\text{Min } \sum_k (\sum_i \sum_j c_{ij} x_{ij}^k) \delta_k$

55.  $\sum_k \delta_k = 1$ ,

56.  $\sum_k \gamma_k = 1$ ,

57.  $\sum_k x_{ij}^k \delta_k - \sum_k x_{ij}^k \gamma_k = 0$ , for all  $i, j$ .

58.  $\gamma_k \geq 0, \delta_k \geq 0$ .

In addition, we wrote an AMPL script (listed in the Appendix) to solve model CS-mflow, with subgradient optimization over  $\beta_{ij}$ . The  $\beta_{ij}$  were updated as  $\beta_{ij} = \beta_{ij} + (x_{ij} - w_{ij}) * \text{stepsize}$ , and  $\text{stepsize}$  was adjusted in a typical manner for subgradient optimization. Again, we found the optimal value of 346.7837104, but not a tour. Two of the  $x_{ij}$  variables were different from their corresponding  $w_{ij}$  variables. When we started

with the dual prices to constraint set 50, we found the optimal tour. In both cases, the new bound was tight.

Example 2. We solved a 10-city example from Boyd & Labonté (2002). They proved that their example has the largest possible gap between the value of the optimal tour of 40 and  $v(\underline{\text{DFJ}})$ , which is 34. For their example, Model CS-mflow gives an optimal value of 36, which is the same objective value as Model BC. Our results suggest that their conjecture may be irrelevant, provided that this new method can be shown to be computationally effective.

Example 3. We solved an example problem from Nemhauser & Wolsey (1988), pages 474. In that example,  $v(\underline{\text{DFJ}}) = 9$ , while  $v(\text{CS-mflow})=10$ , equal to Model BC and the optimal tour length.

The Appendix also gives an AMPL script that writes an SVG graph of the solution, which may be viewed in an SVG-enabled web browser.

## Conclusion and future work

We applied column splitting to the TSP, and showed that it improves on the Held-Karp lower bound. We expect that in fact the bound produced by column splitting equals that of Model BC, that is, the 2-matching problem with the Lagrangean objective.

## References

- Applegate, David, Robert Bixby, Vasek Chvatal, and William Cook, "On the Solution of Travelling Salesman Problems," *Documenta Math.*, vol. 3, pp. 645-656, 1998.
- Claus, A., "A New Formulation for the Travelling Salesman Problem," *SIAM J. Alg. Disc. Math.*, vol. 5, pp. 21-25, 1984.
- Balas, E., and N. Christofides, "A restricted Lagrangean approach to the traveling salesman problem," *Mathematical Programming*, vol. 21, pp. 19-46, 1981. This article also appears in Lawler et al (1985), pp. 378-391..
- Dantzig, G., Fulkerson, R., and Johnson, S., *Solution of a Large-Scale Traveling-Salesman Problem*, *Operations Research* vol. 2, p. 393, 1954.
- Edmonds, J., "Maximum matching and a polyhedron with 0,1-vertices," *J. Res. Nat. Bur. Stand. B*, vol. 69, pp. 125-130, 1965.
- Fourer, R., Gay, D.M., & Kernighan, B.W. (2003) *AMPL: A modeling Language for Mathematical Programming*, Thomson Publishing, Pacific Grove, CA.
- Guignard, Monique, and Kim, Siwhan, "Lagrangean Decomposition: a Model Yielding Stronger Lagrangean Bounds," *Math Programming*, vol. 39, 1987, pp. 215-228.
- Held, Michael, and Karp, Richard M., "The Travelling-Salesman Problem and Minimum Spanning Trees" *Ops Res*, vol. 18, no. 6, Nov-Dec 1970, pp. 1138-1162.
- Held, Michael, and Karp, Richard M., "The Travelling-Salesman Problem and Minimum Spanning Trees: Part II," *Math Programming*, vol. 1, 1971, pp. 6-25.
- Helsgaun, K., "An Effective Implementation of the Lin-Kernighan Travelling Salesman Heuristic," *Datalogiske Skrifter (Writings on Computer Science)*, No. 81, 1998, Roskilde University.
- Lawler, E.L., J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, *The Traveling Salesman Problem: a Guided Tour of Combinatorial Optimization*, John Wiley & Sons, Chichester, 1985.
- Mamer, John W. and McBride, Richard D., "A Decomposition-Based Pricing Procedure for Large-Scale Linear Programs: An Application to the Linear Multicommodity Flow Problem," *Management Science*, vol. 46, no. 5, May 2000.

- Martin, R. Kipp, "Using Separation Algorithms to Generate Mixed Integer Model Reformulations," *Operations Res. Letters*, 10 (1991) 119-128.
- Martin, R. Kipp, Sweeney, Dennis J., and Doherty, Michael E., "The Reduced Cost Branch and Bound Algorithm for Mixed Integer Programming," *Computers & Operations Research*, vol. 12, no. 2, pp. 139-149, 1985.
- Nemhauser, George L, and Wolsey, Lawrence A., *Integer and Combinatorial Optimization*, 1988, John Wiley & Sons, Inc.
- Wong, R.T., "Integer Programming Formulations of the Travelling Salesman Problem," *Proc. IEEE Conf. on Circuits and Computers*, pp. 149-152, 1980.
- Wysocki, Piotr, <http://wysek.thanedd.sk/pub/inf/maxflow.c>, 1993.

## Appendix

AMPL model to find a lower bound on the TSP with column splitting.

```
# You can use this to make a tree, a ltree, or a TSP.
# - To make a tree, use TotalDegree, Stage1, and Stage2.
# Do not have a dummy node in your data.
# - To make a ltree, use Root, TotalDegree, Stage1, and Stage2.
# In your data, copy node 1 to node last+1, to have a dummy node.
# - To solve the TSP, use NodeDegree, Root, Stage1, and Stage2.
# In your data, copy node 1 to last+1, as a dummy node. Make x binary.

param N; set Nodes:= {1..N};
param xcoord {Nodes}; param ycoord {Nodes};
set Arcs := {i in Nodes, j in Nodes: i < j};

param length {i in Nodes, j in Nodes: (i,j) in Arcs}
    := sqrt((xcoord[i] - xcoord[j])^2 + (ycoord[i] - ycoord[j])^2);

param beta {(i,j) in Arcs}; # Dual prices on the Split rows.
var x {(i,j) in Arcs} binary; #>= 0;
var w {(i,j) in Arcs} binary; #>= 0;

set Zset := {i in Nodes, j in Nodes, t in Nodes:
    i <> t and (i,j) in Arcs or (j,i) in Arcs and (t,i) in Arcs};
var z {(i,j,k) in Zset} >= 0;

#minimize Tourlength: sum {(i,j) in Arcs} length[i,j] * x[i,j];
#subject to Split {(i,j) in Arcs}: x[i,j] = w[i,j];

minimize CSrelaxTourlength: sum {(i,j) in Arcs} length[i,j] * x[i,j]
    + sum {(i,j) in Arcs} beta [i,j]*(x[i,j] - w[i,j]);

# All nodes of degree 2, except the first & last, which have degree 1.
# Thus, we have a tour using the model for a 1-tree.
subject to NodeDegree {i in Nodes: i <> 1 and i <> N}:
    sum {(j,i) in Arcs} w[j,i] + sum {(i,j) in Arcs} w[i,j] = 2;

# This constraint prevents duplicating the arc from the depot.
subject to Root {i in Nodes: i >= 2 and i <= N-1}: x[1,i] + x[i,N]<=1;

subject to TotalDegree: sum {(i,j) in Arcs} x[i,j] = card(Nodes) - 1;

subject to Stage1 {(i,j) in Arcs, t in Nodes}:
    (if (i,j,t) in Zset then z[i,j,t]) + (if (j,i,t) in Zset then
z[j,i,t]) = x[i,j];

subject to Stage2 {(t,i) in Arcs}:
    sum {j in Nodes: (i,j) in Arcs or (j,i) in Arcs} z[i,j,t] <= 1;
AMPL script to find a lower bound on the TSP with column splitting.
```

```

# cstspl.run. Find a lower bound on the TSP with column splitting.
option solver cplex; option omit_zero_rows 1, omit_zero_cols 1;
param notequalcount; param stepsize;
model cstspl.mod;
read N, {i in 1..N} (xcoord[i],ycoord[i]) < frac10citygap9813.txt;
solve;

let stepsize := 20;
# Do subgradient optimisation to find the best beta.
for {runjob in 1..1000}
{
  let stepsize := stepsize * .95; let notequalcount := 0;
  for {(i,j) in Arcs}
  {
    if x[i,j] > w[i,j] then
    {
      let beta [i,j] := beta [i,j] + stepsize;
      let notequalcount := notequalcount + 1;
    }
    if x[i,j] < w[i,j] then
    {
      let beta [i,j] := beta [i,j] - stepsize;
      let notequalcount := notequalcount + 1;
    }
  }
  if notequalcount = 0 then break;
  solve;
  printf "%i x's and w's unequal.\n", notequalcount;
}
commands _makesvg.txt;

```

Script `_makesvg.txt`. From the output to the TSP model, this file makes an SVG graph of the solution, which may be viewed in an SVG-enabled web browser.

```

# Creates SVG file tspgraph.svg. Assume N nodes, xcoord, ycoord, x.
# Print SVG file header.
printf "<?xml version="1.0" standalone="yes"?>\n<!DOCTYPE svg
PUBLIC "-//W3C//DTD SVG 20010904//EN"
"http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">\n<svg
viewBox="0 0 %i %i" xmlns="http://www.w3.org/2000/svg">\n<g>,
1.05*max{i in Nodes} xcoord[i], 1.05*max{i in Nodes} ycoord[i] >
tspgraph.svg;

# Sizes for objects.
param textsize;
let textsize := max(max{i in Nodes} xcoord[i], max{i in Nodes}
ycoord[i])/100;

for {(i,j) in Arcs: x[i,j] > 0.001}
  printf "<line xl="g" yl="g" x2="g" y2="g"
stroke="black" stroke-width="g"/>\n",
  xcoord[i], ycoord[i], xcoord[j], ycoord[j], textsize/8 >>
tspgraph.svg;

for {i in Nodes} # Print circles on the nodes.
{
  printf "<circle cx="g" cy="g" r="g" fill="none"
stroke="gray" stroke-width="g"/>\n",
  xcoord[i], ycoord[i], 0.4*textsize, textsize/4 >> tspgraph.svg;
  # Print labels in the nodes.
  printf "<text x="g" y="g" font-family="Verdana" font-
size="g" fill="blue">%i</text>\n",
  xcoord[i]-0.2, ycoord[i]+0.1, 2.5*textsize, i >> tspgraph.svg;
}
printf "</g></svg>" >> tspgraph.svg; close tspgraph.svg;

Sample data file.

```

```

# Data file: N   x1 y1  x2 y2   x3 y3   ...
# Tour length 346.7837104, v(HK) = 340.3072057.
10  0 0  10 69  35 97  40 51  54 58  56 28  65 80  66 86  74 30  95 45

```