A NEW LAGRANGIAN RELAXATION BASED BRANCH AND BOUND ALGORITHM FOR THE TRAVELING SALESMAN PROBLEM

Cenk Çalışkan, Woodbury School of Business, Utah Valley University, 800 W. University Pkwy, Orem, UT 84058, (801) 863-6487, cenk.caliskan@uvu.edu

ABSTRACT

We study the asymmetric Traveling Salesman Problem (TSP) which can be defined as follows: a salesman has to visit n cities exactly once and minimize the total cost of the trip in doing so, where the travel costs between a pair of cities may be different in each direction. The TSP is NP-complete, so no efficient solution algorithm exists. We propose a promising new algorithm that is based on Lagrangian relaxation, subgradient optimization and branch-and-bound methods. Our integer programming formulation reduces to a shortest path problem when tour enforcing constraints are relaxed. Thus, we only solve a series of shortest path problems on an acyclic network. This, together with tight Lagrangian lower bounds makes our method particularly attractive.
Keywords: traveling salesman problem, lagrangian relaxation, shortest path problem, subgradient optimization

INTRODUCTION

The Traveling Salesman Problem (TSP) is an important problem in management science. The statement of the problem is simple: a traveling salesman has to visit n cities exactly once on a trip of minimum cost, where there is a known cost to go from one city to another. The asymmetric TSP is the version of the problem in which the cost may be different between a pair of cities in each direction. The TSP is NP-Complete (Garey and Johnson [11]). This means that there are no polynomial time solution algorithms. The TSP was first introduced by Dantzig et al. [6] in which an assignment based integer programming (IP) formulation was proposed. An exponential number of sub-tour elimination constraints (SECs) are used in this model to prevent partial tours. Most of the formulations of the TSP use the assignment problem as their core, like in this earliest model. We base our model on a shortest path core rather than an assignment one. Little at al. [20] propose the first branch and bound algorithm for the TSP that is based on the assignment formulation. Held and Karp [16] propose a branch and bound algorithm based on the formulation of Dantzig et al. [6], but takes advantage of the fact that a TSP tour is a 1-tree, i.e. a spanning tree with an additional arc. Miller et al. [22] introduce the first polynomial size formulation of the TSP. They use auxiliary variables similar to node potentials to prevent subtours. Their SECs are compact and they are used widely in formulating the extensions of the TSP. Fox et al. [10] propose another polynomial size formulation for the TSP that uses $O(n)$ constraints. Balas and Christofides [3] propose a Lagrangian relaxation based branch and bound algorithm for the formulation of Dantzig et al. [6], which also uses the concept of adding valid inequalities or cuts. Claus [5] proposes another polynomial size formulation with $O(n^3)$ constraints, with the use of auxiliary variables. Noon and Bean [23] propose another Lagrangian relaxation based branch and bound procedure for the generalized TSP with an assignment core. Langevin et al. [17] compare and show the relationships between different formulations of the TSP including the standard formulation of Dantzig et al. [6] and the formulation by Miller et al. [22]. Gavish and Srijkanth [12] study the multiple TSP and propose

another Lagrangian relaxation based branch and bound method that is based on the original Dantzig et al. [6] formulation. Padberg and Sung [26] compare four different formulations of the TSP and find that the standard formulation is the strongest of them all in terms of LP relaxation bounds. Desrochers and Laporte [7] extend and improve the SECs proposed by Miller et al. [22]. Shapiro [27] propose the same shortest path based model that we propose in this paper and develop a Lagrangian relaxation based solution procedure. However, the proposed approach is not practical to implement as an efficient algorithm and for that reason, there are no computational results in the literature to date. Fischetti and Toth [8] propose a branch and bound procedure based on shortest spanning arborescence problem for the standard assignment based formulation. Balas [2] proposes some polynomially solvable special cases of the TSP. Langevin et al. [18] propose a two-commodity flow based formulation that has $O(n)$ constraints and $O(n^2)$ variables. Gouveia and Voß [15] compare six formulations of the time-dependent TSP (TDTSP) problem and show which of them are stronger formulations. Lysgaard [21] propose a new cluster based branching scheme for the standard formulation, whereas Gouveia and Pires [13] [14] propose a new reformulation of the Miller et al. [22] model. Sherali and Driscoll [28] propose further improvements to the Miller et al. [22] SEC model. Fischetti et al. [9] propose a branch and cut procedure for the standard formulation. Orman and Williams [25] survey eight different formulations of the TSP and compare the sizes of their polytopes as well as the strength of their LP relaxations. Sherali et al. [29] propose yet another class of polynomial size formulations for the TSP that have tighter LP relaxations than the standard formulation. Öncan et al. [24] provide a comprehensive survey of 24 different formulations of the asymmetric TSP, whereas Laporte [19] provides a concise guide to the TSP problem and summarizes the proposed formulations and algorithms to date. Bektaş and Gouveia [4] propose a generalization of the SECs first proposed by Miller et al. [22].

In this paper, we propose a computationally attractive Lagrangian relaxation based branch and bound procedure on a shortest path based formulation. The solution procedure involves solving a series of progressively smaller shortest path problems on an acyclic network, which can be efficiently solved in practice as the computational effort for such shortest path problems is $O(nm)$ on the extended network that we propose.

PROBLEM DESCRIPTION

Let $G = (N, A)$ be a directed network consisting of a set N of nodes and a set A of arcs, where $|N| = n$ and $|A| = m$. In this network, nodes i and j represent two different cities and arc (i, j) represents the one-way travel lane from i to j. The traveling cost on arc (i, j) is c_{ij}. The traveling salesman problem (TSP) is then to find a minimum cost Hamiltonian cycle on G, i.e. a tour or cycle that visits each node exactly once, assuming one exists.

We replicate each node in N exactly n times and denote the resulting node set N'. We order the copies from 1 to n and denote copy t of node i by $i(t)$. We then create an arc $(i(t), j(t+1))$ for each arc (i, j) in G, for $t = 1, 2, \ldots, n-1$ and an arc $(i(t), j(1))$ for $t = n$. We denote the resulting arc set A' and the resulting extended network $G' = (N', A')$. The extended network is shown in Figure 1. A Hamiltonian cycle on G corresponds to a cycle in G' that visits exactly one copy of each node, traversing the outside of the cylinder in Figure 1. The cycle that has the minimum total cost among all such cycles solves the TSP.

We can arbitrarily fix one of the nodes to the first position in the cycle, say, node 1. We then remove from G' nodes $1(t)$ for $t = 2, 3, \ldots, n$ and nodes $i(1)$, $i \neq 1$ as they become redundant. We then create a

node 1(n+1) and replace all arcs (i(n), 1(1)) with (i(n), 1(n+1)) with the same cost, i.e. c_{i1}. That transforms the extended network to an acyclic network as shown in Figure 2. A Hamiltonian cycle on G is now a path from 1(1) to 1(n+1) that visits only a copy of nodes 2, 3, … , n exactly once. Thus, the TSP becomes a constrained shortest path problem on the modified extended network.

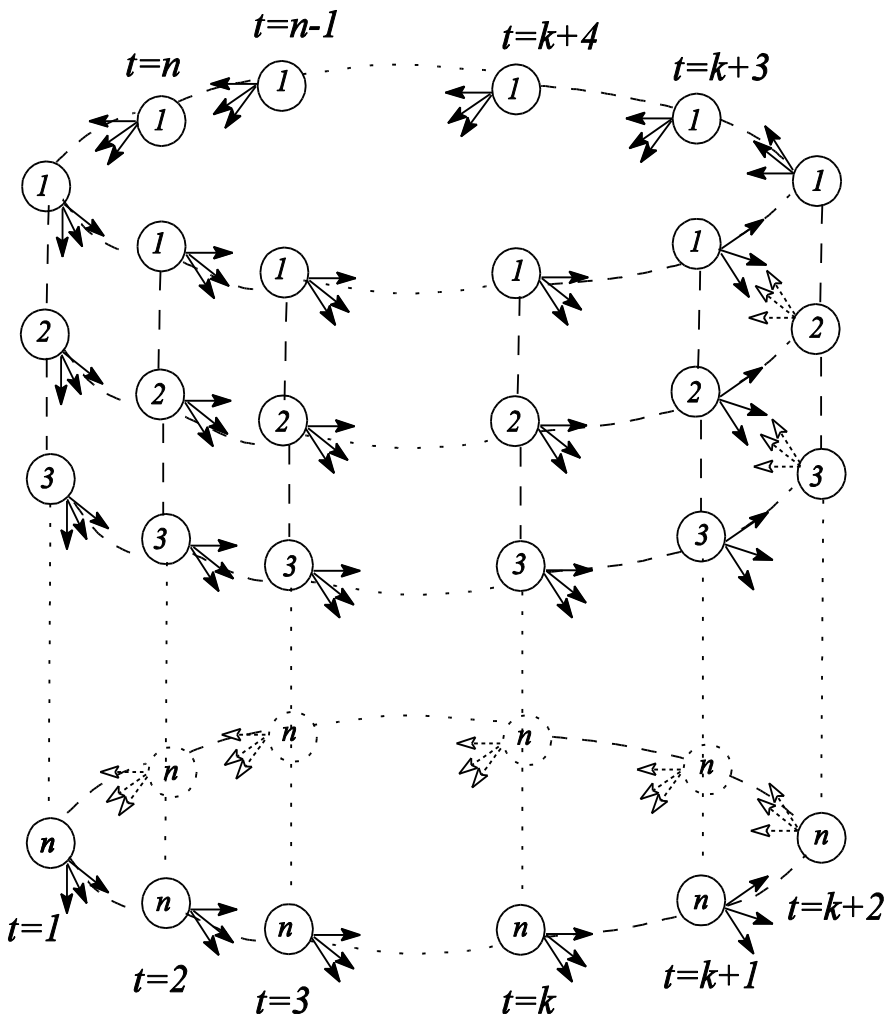

Figure 1: The extended network

We define a flow variable $x_{i(t)\,j(t+1)}$ for each arc in A'. We then formulate the TSP on the extended network G' as a constrained network flow problem in which a flow of 1 units is sent from node 1(1) to 1(n+1) subject to the constraints that the flows coming into each node i(t) for t = 2, 3, … , n+1 add up to 1 for i = 1, 2, 3, … , n and all of the flows are integral. These constraints ensure that each node i is visited exactly once in any feasible solution, making it a Hamiltonian cycle or a feasible TSP tour in the original network. The minimization of this integer program results in an optimal solution to the TSP. Thus, the integer programming formulation of the TSP on the modified extended network can be stated as follows:

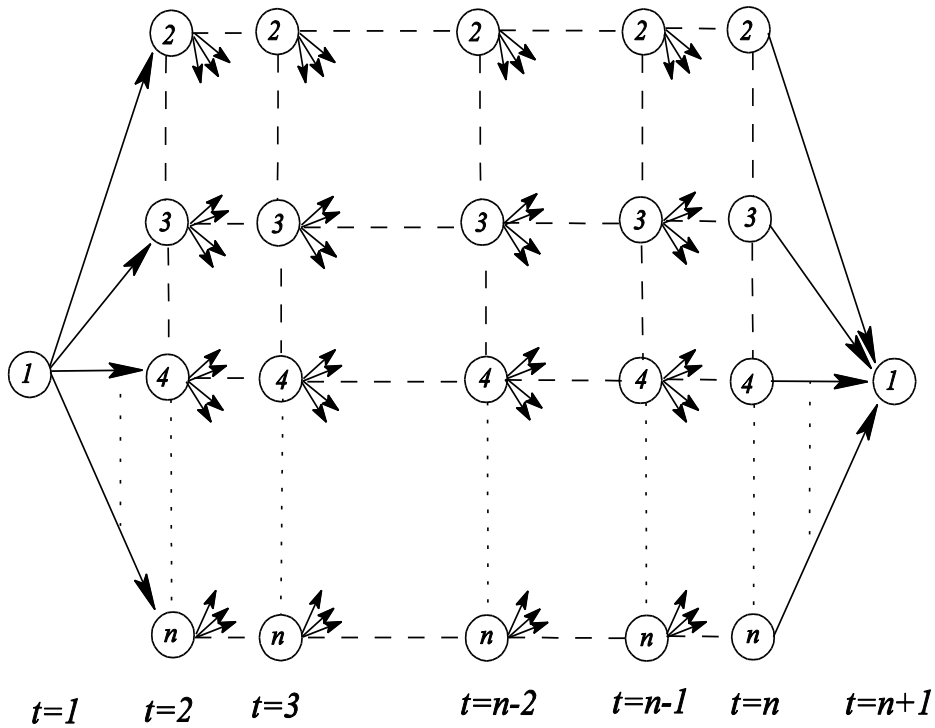

Figure 2: The modified extended network

$$[P] \min \sum_{t=1}^{n} \sum_{(i,j)\in A} c_{ij} x_{i(t)j(t+1)} \tag{1}$$

s.t.

$$\sum_{(i,j)\in A} x_{i(t)j(t+1)} - \sum_{(j,i)\in A} x_{j(t-1)i(t)} = 0 \quad \forall i(t) \in N' \setminus \{1(1), 1(n+1)\} \tag{2}$$

$$\sum_{t=2}^{n} \sum_{(j,i)\in A} x_{j(t-1)i(t)} = 1 \quad \forall i \in N \setminus \{1\} \tag{3}$$

$$x_{i(t)j(t+1)} \geq 0 \quad \textit{and integer} \quad \forall (i(t), j(t+1)) \in A' \tag{4}$$

Eq. 1 is the total cost of a feasible flow in the extended network, or the total cost of a feasible TSP tour. Eqs. 2 are the conservation of flow constraints for all the intermediate nodes in the network. Eqs. 3 are the tour enforcing constraints and Eqs. 4 are the nonnegativity and integrality constraints. Note that we did not explicitly define conservation of flow constraints for the source and sink nodes, i.e. for nodes 1(1) and 1(n+1) as Eqs. 3 and 4 ensure that only 1 unit of flow will leave the source and arrive at the sink.

LAGRANGIAN RELAXATION METHOD

If we relax the tour enforcing constraints (Eqs. 3), the resulting problem is a shortest path problem on a directed acyclic graph. The computational effort to find a shortest path on such a graph is well known to be $O(m)$. In G', the number of arcs is nm, so the computational effort to find a shortest path is $O(nm)$.

This makes the Lagrangian relaxation approach particularly attractive computationally. We define the Lagrange multipliers λ_i, for $i = 2,3,...,n$ corresponding to the tour enforcing constraints (Eqs. 3). Let X be the set of x that satisfy Eqs. 2 and 4. We then relax the tour enforcing constraints and bring them into the objective function, resulting in the following Lagrangian subproblem or Lagrangian function:

$$L(\lambda) = \min\{\sum_{t=1}^{n}\sum_{(i,j)\in A} c_{ij}x_{i(t)j(t+1)} + \sum_{i=2}^{n}\lambda_i(\sum_{t=2}^{n}\sum_{(j,i)\in A} x_{j(t-1)i(t)} - 1) : x \in X\}.$$

This can be simplified into the following equation and it is obviously a shortest path problem for a given vector of values for the Lagrange multipliers:

$$L(\lambda) = \min\{\sum_{t=1}^{n}\sum_{(i,j)\in A}(c_{ij} + \lambda_j)x_{i(t)j(t+1)} - \sum_{i=2}^{n}\lambda_i : x \in X\} \tag{5}$$

Let $\delta_i = \sum_{t=2}^{n}\sum_{(j,i)\in A} x_{j(t-1)i(t)} - 1$ for all $i \in N \setminus \{1\}$. So, δ_i represents the number of extra visits to node i in excess of 1 on a tour corresponding to a flow vector x. Then, Eq. 3 could be re-written in the form $\delta = 0$.

Lemma 1. *For any value λ of the Lagrange multipliers, the value $L(\lambda)$ of the Lagrangian function is a lower bound on the optimal objective function value z^* of the original problem, P.*

Proof: Since for every feasible $x \in X$, each node is visited exactly once in the corresponding tour and thus $\delta = 0$, for any value of Lagrange multipliers λ. Therefore,

$$\begin{aligned} z^* &= \min\{\sum_{t=1}^{n}\sum_{(i,j)\in A} c_{ij}x_{i(t)j(t+1)} : \delta = 0, x \in X\} \\ &= \min\{\sum_{t=1}^{n}\sum_{(i,j)\in A} c_{ij}x_{i(t)j(t+1)} + \sum_{i=2}^{n}\lambda_i\delta_i : \delta = 0, x \in X\} = L(\lambda). \end{aligned}$$

Since removing the constraint $\delta = 0$ from the second equation cannot increase the optimal value of $L(\lambda)$, then we have $z^* \geq L(\lambda)$.

In order to obtain the tightest lower bounds, we maximize the Lagrangian function, i.e. we solve the following maximization problem, which is called the Lagrangian dual:

$$L^* = \max_{\lambda} L(\lambda). \tag{6}$$

In order to solve the Lagrangian dual, we use a subgradient optimization procedure. We use an adaptation of the Newton's method in which we update the Lagrange multipliers in a relatively large step towards the optimal solution along the subgradient direction. The initial value of the multipliers are set to zero, and at every iteration, they are updated as follows:

$$\lambda_i^{k+1} = \lambda_i^k + \theta_k \delta_i \quad \forall i \in N \setminus \{1\} \tag{7}$$

where θ_k is the step size at iteration k. So, λ is not modified for the nodes that are visited only once, is increased for the ones that are visited more than once, and decreased for the ones that are not visited. Intuitively, the procedure makes the nodes that are not visited more attractive by reducing their visiting costs and makes the ones that are visited too many times unattractive by increasing their costs, while not changing the ones that are visited only once.

In order to make sure that the procedure will converge, the step size is chosen as follows (see, for instance, Ahuja et al. [1]):

$$\theta_k = \frac{\sigma_k [UB - L(\lambda^k)]}{\| \delta \|^2} \tag{8}$$

where $\| \delta \| = (\sum_i \delta_i^2)^{0.5}$ is the Euclidean norm of δ and σ_k is a scalar that is chosen strictly between 0 and 2. the initial value of σ is 2, and it is halved once the Lagrangian objective function fails to increase after a long series of iterations. UB is the surrogate for the optimal Lagrangian objective function value, L^* and it is chosen as the best upper bound, i.e. the total cost of the lowest cost feasible TSP tour found thus far in the algorithm.

procedure *solve_lagrangian_dual(*Γ*)*;
begin
  $z := \sum_{(i,j) \in P} c_{ij}$;
  $\lambda := 0$;
  $\sigma := 2$;
  $count := 0$;
  while P is not a TSP tour **and** $z < UB$ **and** $count < T_{max}$ **do**
    $\theta := \sigma [UB - L(\lambda)] / \| \delta \|^2$;
    $\lambda := \lambda + \theta \delta$;
    $c_{ij} := c_{ij} + \lambda_j \quad \forall j \in \Gamma \setminus \{1\}$;
    P := Shortest path from $1(1)$ to $1(n+1)$ in Γ;
    $z := \sum_{(i,j) \in P} c_{ij}$;
    $count := count + 1$;
    if z has not changed in T_{lim} iterations **then**
      $\sigma := \sigma / 2$;
    end if
  end while
  return (z, P);
end

Figure 3: The procedure *solve_lagrangian_dual* of the branch and bound algorithm

The pseudocode for the subgradient method to solve the Lagrangian dual problem is given in Figure 3. Note that the iterations of the subgradient method may be stopped whenever $L(\lambda) \geq UB$ as that means

we cannot obtain a better feasible tour than the best existing tour from this branch and bound node. We describe the branch and bound algorithm in the next section. The pseudocode for the branch and bound algorithm is shown in Figure 4.

THE BRANCH AND BOUND ALGORITHM

We start the branch and bound procedure by solving the Lagrangian dual for the original problem using the subgradient method. If the resulting solution is a feasible tour, we stop. We found the optimal TSP tour. If the resulting solution is not a TSP tour, then one or more nodes are visited more than once and others are not visited at all. We then branch on one of the nodes that are visited more than once. We choose the node that is visited most and fix that node to one of the positions in the tour. This can easily be done by removing all of the other nodes at that position (i.e. at that t). We pick the position with the lowest cost of visit. The other branch is where we constrain that node not to be visited at that same position t. This can easily be accomplished by removing the node from that position. The two branches are mutually exclusive.

```
algorithm   branch_and_bound;
begin
    UB := ∞ ;
     P := Shortest path from 1(1) to 1(n+1) in G'
     if P is a TSP tour then
         Stop. P is optimal and z* := Σ_{(i,j)∈P} c_ij;
     else
          B = ∅ ;
          make_branches;
         P :=  make_feasible_heuristic;
         UB :=  min{UB, Σ_{(i,j)∈P} c_ij} ;
         while B ≠ ∅ and Ĝ' := a branch and bound node from B do
             { LB , P } :=  solve_lagrangian_dual(Ĝ');
             if P is a TSP tour then
                 UB := min{UB, Σ_{(i,j)∈P} c_ij} ;
                 B = B \ Ĝ';
             else
                 if LB < UB then
                     make_branches;
                     P :=  make_feasible_heuristic;
                     UB := min{UB, Σ_{(i,j)∈P} c_ij} ;
                 end if
                 B = B \ Ĝ';
             end if
         end while
    end if
end
```

Figure 4: The algorithm *branch_and_bound*

We then apply the subgradient method to one of the branches that we pick and keep repeating this until all of the created branches are fathomed at which time we will have found the optimal TSP tour. If any of the branches has a lower bound greater than the cost of the best feasible tour, that branch is fathomed because it cannot yield a better solution. If any of the branches yields a feasible tour, then that branch is fathomed as well, as we can't find a better solution from that branch. We update the best UB if this feasible solution has a lower total cost. We apply a heuristic to each of the infeasible tours created by the subgradient method and convert it to a feasible tour. The pseudocode of the heuristic is given in Figure 5.

procedure *make_feasible_heuristic*;
begin
  $\hat{G}' := G'$;
  Let P be the path from $1(1)$ to $1(n+1)$ in $\hat{G}'$;
  while P is not a TSP tour **do**
    for every node $i \in N \setminus 1$ **do**
      if $\delta_i = 1$ **then**
        Remove from $\hat{G}'$ every node $i(t)$ for which $x_{j(t)i(t+1)} = 0$;
      else if $\delta_i > 1$ **then**
        $\hat{t} := argmin_t(c_{ji} + c_{ij}, \forall x_{j(t)i(t+1)} = 1)$;
        Remove from $\hat{G}'$ every node $i(t)$ for which $t \neq \hat{t}$;
      end if
    end for
    P := Shortest path from $1(1)$ to $1(n+1)$ in $\hat{G}'$
  end while
  return P;
end

Figure 5: The procedure *make_feasible_heuristic* of the branch and bound algorithm

procedure *make_branches(P)*;
begin
  $\hat{i} := argmax_i(\delta_i, \forall i \in P)$;
  $\hat{t} := argmin_t(c_{j\hat{i}} + c_{\hat{i}j}, \forall x_{j(t)\hat{i}(t+1)} = 1)$;
  $G_1' := G'$;
  $G_2' := G'$;
  { *Create branch 1*}
  Remove from G_1' every node $\hat{i}(t)$ for which $t \neq \hat{t}$;
  $\mathsf{B} := \mathsf{B} \cup \{G_1'\}$
  { *Create branch 2*}
  Remove from G_2' node $\hat{i}(\hat{t})$;
  $\mathsf{B} := \mathsf{B} \cup \{G_2'\}$

end

Figure 6: The procedure *make_branches* of the branch and bound algorithm

CONCLUSION

In this research we propose a constrained shortest path based integer programming formulation for the Traveling Salesman Problem (TSP). We also propose a Lagrangian relaxation based branch and bound procedure to solve the TSP. At each node of the branch and bound tree, subgradient optimization method is applied to solve the Lagrangian dual problem to obtain a lower bound. The proposed method is especially attractive as the entire solution process reduces to a series of shortest path problems on acyclic networks, which can be solved efficiently in $O(nm)$ time. It is also attractive due to the fact that Lagrangian relaxation lower bounds are typically tighter than LP relaxation lower bounds. We propose a simple heuristic to convert infeasible tours created by the solutions to the Lagrangian dual problems to TSP tours. We describe the details of the algorithm in this paper. Future research will focus on computational testing of the proposed formulation and the solution procedure as compared to the application of the branch and bound method to other formulations of the TSP.

REFERENCES

[1] Ahuja, R.K., Magnanti, T.L. & Orlin, J.B. *Network flows: Theory, algorithms and applications*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[2] Balas, E. New classes of efficiently solvable generalized traveling salesman problems. *Annals of Operations Research*, 1999, 86:529–558.

[3] Balas, E. & Christofides, N. A restricted Lagrangean approach to the traveling salesman problem. *Mathematical Programming*, 1981, 21(1):19–46.

[4] Bektaş, T. & Gouveia, L. Requiem for the Miller-Tucker-Zemlin subtour elimination constraints? *European Journal of Operational Research*, 2014, 236:820–832.

[5] Claus, A. A new formulation for the travelling salesman problem. *SIAM Journal on Algebraic and Discrete Methods*, 1984, 5(1):21–25.

[6] Dantzig, G.B., Fulkerson, D.R. & Johnson, S.M. Solution of a large-scale traveling-salesman problem. *Operations Research*, 1954, 2:393–410.

[7] Desrochers, M. & Laporte, G. Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. *Operations Research Letters*, 1991, 10(1):27–36.

[8] Fischetti, M. and Toth, P. An additive bounding procedure for the asymmetric travelling salesman problem. *Mathematical Programming*, 1992, 53(1-3):173–197.

[9] Fischetti, M., Lodi, A. & Toth, P. *Solving Real-World ATSP Instances by Branch-and-Cut*, volume 2570 of *Lecture Notes in Computer Science*, Berlin Heidelberg: Springer-Verlag, 2003, 64–77.

[10] Fox, K.R., Gavish, B. & Graves, S.C. An n-constraint formulation of the (time-dependent) traveling salesman problem. *Operations Research*, 1980, 28(4):1018–1021.

[11] Garey. M.R. & Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY: W.H. Freeman and Co., 1979.

[12] Gavish, B. & Srikanth, K. An optimal solution method for large-scale multiple traveling salesmen problems. *Operations Research*, 1986, 34(5):698–717.

[13] Gouveia, L. & Pires, J.M. The asymmetric travelling salesman problem and a reformulation of the Miller-Tucker-Zemlin constraints. *European Journal of Operational Research*, 1999, 112:134–146.

[14] Gouveia, L. & Pires, J.M. The asymmetric travelling salesman problem: on generalizations of

disaggregated Miller-Tucker-Zemlin constraints. *Discrete Applied Mathematics*, 2001, 112:129–145.

[15] Gouveia, L. & Voß, S. A classification of formulations for the (time-dependent) traveling salesman problem. *European Journal of Operational Research*, 1995, 83:69–82.

[16] Held, M. & Karp, R.M. An algorithm for the traveling salesman problem. *Operations Research*, 1970, 18(6):1138–1162.

[17] Langevin, A., Soumis, F. & Desroisers, J. Classification of travelling salesman problem formulations. *Operations Research Letters*, 1ssssss990, 9(2):127–132.

[18] Langevin, A., Desrochers, M., Desrosiers, J., Gélinas, S. & Soumis, F. A two commodity flow formulation for the traveling salesman and the makespan problems with time windows. *Networks*, 1993, 23(7):631–640.

[19] Laporte, G. A concise guide to the traveling salesman problem. *Journal of the Operational Research Society*, 2010, 61:35–40.

[20] Little, J.D., Murty, K.G., Sweeney, D.W. & Karel, C. An algorithm for the traveling salesman problem. *Operations Research*, 1963, 11(6):972–989.

[21] Lysgaard, J. Cluster based branching for the asymmetric traveling salesman problem. *European Journal of Operational Research*, 1999, 119:314–325.

[22] Miller, C.E., Tucker, A.W. & Zemlin, R.A. Integer programming formulation of traveling salesman problems. *Journal of the ACM*, 1960, 7(4):326–329.

[23] Noon, C.E. & Bean, J.C. A Lagrangian based approach for the asymmetric generalized traveling salesman problem. *Operations Research*, 1991, 39(4):623–632

[24] Öncan, T., Altınel, İ.K. & Laporte, G. A comparative analysis of several asymmetric traveling salesman problem formulations. *Computers and Operations Research*, 2009, 36:637–654.

[25] Orman, A.J. & Williams, H.P. A survey of different integer programming formulations of the travelling salesman problem. Working paper, London School of Economics and Political Science, 2004.

[26] Padberg, M. & Sung, T.Y. An analytical comparison of different formulations of the travelling salesman problem. *Mathematical Programming*, 1991, 52(1-3):315–357.

[27] Shapiro, J.F. Convergent duality for the traveling salesman problem. *Operations Research Letters*, 1991, 10(3):129–136.

[28] Sherali, H.D. & Driscoll, P.J. On tightening the relaxations of Miller-Tucker-Zemlin formulations for asymmetric traveling salesman problems. *Operations Research*, 2002, 50(4):656–669.

[29] Sherali, H.D., Sarin, S.C.& Tsai, P.F. A class of lifted path and flow-based formulations for the asymmetric traveling salesman problem with and without precedence constraints. *Discrete Optimization*, 2006, 3:20–32.