2009-05-01

Tight Lower Bound for the Sparse Travelling Salesman Problem

Fredrick Mtenzi
Technological University Dublin, Fredrick.Mtenzi@tudublin.ie

Recommended Citation

Mtenzi, F (2009). Tight Lower Bound for the Sparse Travelling Sale. *ICIT 2009: 4th. International Conference on Information Technology*, Amman, Jordan. doi:10.21427/g6b0-0y08

Tight Lower Bound for the Sparse Travelling Salesman Problem

Fredrick Mtenzi
School of Computing, Computer Science Department,
Dublin Institute of Technology, Dublin 8, Ireland
Fredrick.Mtenzi@dit.ie

ABSTRACT

The Sparse Travelling Salesman Problem (Sparse TSP) which is a variant of the classical Travelling Salesman Problem (TSP) is the problem of finding the shortest route of the salesman when visiting cities in a region making sure that each city is visited at least once and returning home at the end. In the Sparse TSP, the distance between cities may not obey the triangle inequality; this makes the use of algorithms and formulations designed for the TSP to require modifications in order to produce near-optimal results.

A lower bound for optmisation problems gives us the quality guarantee of the near-optimal solution obtained by using heuristic methods. In this paper we propose two methods of finding tight lower bound for the Sparse TSP. The first method uses the integer linear programming relaxation for the Sparse TSP and the Embedded Flow Formulation (EFF) for the Sparse TSP. The second method proposes a strategy for quickly generating some of the violated arc-cutset constraints which we call an Arc-cutset Partial Enumeration Strategy (APES).

Key Words: Sparse TSP, LP relaxation, tight lower bound, formulation, arc-cutset constraint

1 1. Introduction

When solving the Sparse TSP, our main interest is in computing feasible tours at a reasonable computational cost. In addition because it is not possible to get an optimal solution most of the time, we would like to have some guarantee on the quality of the tours (solutions) found. Such guarantees can most of the time be provided if a lower bound on the length of a shortest possible tour is known.

It is also the case that most algorithms for finding exact solutions for large Standard TSP instances are based on methods for finding upper and lower bounds and an enumeration scheme. For a given instance, lower and upper bounds are computed. In most cases, these bounds will not be equal, and therefore, only a quality guarantee for the feasible solution can be given, but optimality cannot be proved. If the upper and lower bounds coincide, a proof of optimality is achieved.

Therefore, the determination of good tours and derivation of tight lower bounds can be keys to a successful search for optimal solutions. Whereas there have been a lot of work and progress in designing heuristic methods to produce upper bound, the situation for lower bounds is not as satisfying.

In general for the Standard TSP, lower bounds are obtained by solving relaxations of the original problem in the sense that one optimizes over some set containing all feasible solutions of the original problem as a (proper) subset. This then means, for example, that the optimal solution of the relaxed problem gives a lower bound for the value of the optimal solution of the original problem. In practice, the methods usually used for computing lower bound for the Standard TSP are the Held-Karp lower bound [1] and Lagrangian relaxation [2].

Since the Sparse TSP is an NP-Hard combinatorial optimization problem as per Fleischmann [3], the standard technique to

solve it to optimality is based on an enumeration scheme which for large problems is computationally expensive. Therefore a natural way is to use the Sparse TSP heuristics to obtain a near-optimal solution. Solutions obtained by heuristics for the Sparse TSP provide the upper bounds. Heuristics produce feasible solutions but without any quality guarantees as to how far off they may be from the optimal feasible solution. In order to be able to assess the performance of heuristics we need to find the lower bound of the problem.

Therefore, in this paper we are interested in exploring methods for computing lower bounds for the Sparse TSP. This is the case because we do not have the luxury of comparing with what other researchers have done, since most of the work in the TSP has been focused on the Standard TSP. For example, in the Standard TSP there are sample instances with optimal solutions provided in the TSPLIB (see Reinelt [4]) for most of the problems. The results given in TSPLIB include a provable optimal solution if available or an interval given by the best known lower bound and upper bound. As far as we are aware there are no such benchmark results for the Sparse TSP which is studied in this paper.

A lower bound gives us the quality guarantee of the near-optimal solution obtained by using heuristic methods. The most widely used procedure for finding the lower bound for the Standard TSP is the Held and Karp lower bound [5]. Johnson et al [1] provide empirical evidence in support of using the Held and Karp (HK) lower bound as a stand-in for the optimal tour length when evaluating the quality of near-optimal tours. They show that for a wide variety of randomly generated instances the optimal tour length averages less than 0.8% over the HK lower bound, and for the real world instances in TSPLIB the gap is always less than 2%. A tight lower bound for the Sparse TSP will play a key role in developing and assessing the performance of Sparse TSP heuristic methods.

Some of the definitions we are going to use in this paper.
A *relaxation* of an optimization problem P is another optimization problem R, whose set of feasible solutions $\Re$ properly contains all feasible solutions P of P. The objective function of R is an arbitrary extension on $\Re$ of the objective function of P. Consequently, the objective function value of an optimal solution to R (minimization case) is less than or equal to the objective function value of an optimal solution to P. If P is a hard combinatorial problem and R can be solved efficiently, the optimal value of R can be used as a lower bound in an enumeration scheme to solve P. The closer the optimal value of R to the optimal value of P, the more efficient is the enumeration algorithm.

A *lower bound* of the TSP is the value obtained by solving a relaxation of the original problem or by using heuristics. Its value is in most cases less than the optimal value of the original problem, it is equal to optimal value when the value of lower bound is equal to the value of the upper bound.

The rest of the paper is organised as follows. In this paper, we consider three methods for finding lower bounds for the Sparse TSP. In section 2, we discuss methods for finding the lower bound of the Sparse TSP. The formulation and relaxation for the LP relaxation of the Sparse TSP are covered in section 3. In section 4 we introduce the Arc-cutset Partial Enumeration Strategy as a strategy for finding the lower bound of the Sparse TSP. Finally, section 4 gives the summary and proposes future work in the area.

2 Methods for finding Lower bound for the Sparse TSP

The standard technique for obtaining lower bounds on the Standard TSP is to use a relaxation that is easier to solve than the

original problem. These relaxations can have either discrete or continuous feasible sets. Several relaxations have been considered over years for the Standard TSP. We are going to introduce modifications to these relaxations, so that they can be used to find lower bounds for the Sparse TSP at a reasonable computational effort.

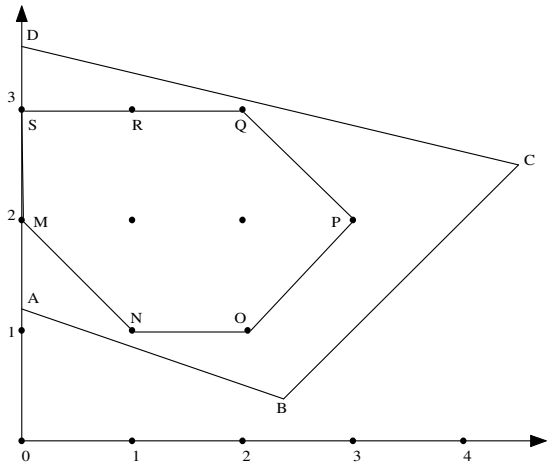

Figure 1: Region MNOPQRS shows a feasible solution to the original `true' problem while region ABCD shows a feasible solution to the relaxed problem

We can illustrate the relaxation of an optimization problem by figure 1 in which the region *ABCD* is the feasible region of the relaxed problem, while the region *MNOPQRS* is the feasible region of the original `true' problem, which in this case happens to contain integer points. Solution *ABCD* may be obtained when we relax integrality constraints. Relaxation of the combinatorial optimization has become so popular because in most cases the problems can be solved with reasonable computational effort, if not easier than the original problem.

The HK lower bound is the solution to the LP relaxation of the integer programming formulation of the Standard TSP (see Dantzig et al [6], Reinelt [2] and Johnson et al [1]). That is, it is the Integer Linear Programming with the integrality constraints relaxed. The HK lower bound provides a very good estimate of optimal tour length for the Standard TSP. This measure has enormous practical value when evaluating the quality of near optimal solutions for large problems where the true optimal solutions are not known or are computationally expensive to find. The HK lower bound has been used as a stand-in for the optimal tour length when evaluating the quality of near-optimal tours in a lot of studies (for example, in Johnson et al [1]).

Although the HK lower bound can be evaluated exactly by Linear Programming techniques, code for doing this efficiently for problems larger than a few hundred cities is not readily available or easy to produce (see Valenzuela and Jones [7]). In addition linear programming implementations (even efficient ones) do not scale well and rapidly become impractical for problems with many thousands of cities. To be able to find the HK lower bound, a procedure for finding violated inequalities must be provided. This is not a simple matter of automatically generating violated inequalities. It is because of the above mentioned difficulties that most researchers have preferred to use the iterative estimation approach for finding lower bound for the Standard TSP proposed by Held and Karp [5],[8]. In this paper we use this method and modify it to solve the Sparse TSP problems.

3 The LP Relaxations for the Sparse TSP

3.1 The LP Relaxations for the Sparse TSP

The formulation for the ILP Sparse TSP is given as:

$$\min_x \sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij}x_{ij} \qquad (1.1)$$

subject to

$$\sum_{j:(i<j)\in A} x_{ij} + \sum_{j:(j<i)\in A} x_{ji} = 2m_i, \text{for all } i \in N \qquad (1.2)$$

$$\sum_{i\in S, j\in N-S, i<j} x_{ij} + \sum_{i\in S, j\in N-S, j<i} x_{ji} \geq 2, \quad \text{for all } S \subset N, S \neq \varnothing \qquad (1.3)$$

$$m_i \geq 0 \text{ for all } i \in N \qquad (1.4)$$

$$x_{ij} \in \{0,1\} \text{ for all } i, j = 1, \cdots, N \qquad (1.5)$$

By relaxing the integrality constraint (1.5) we get the LP relaxation for the ILP Sparse TSP where equations (1.1, 1.2, 1.3 and 1.4) remains the same and equations (1.5) becomes.

$$x_{ij} \geq 0 \text{ for all } i, j = 1, \cdots, N \qquad (1.6)$$

Note: that any integral solution to the LP relaxation is a tour.

We solved the LP relaxation for the modified ILP Sparse TSP formulation problems, results given in figure 7 were obtained by using violated arc-cutset constraints which were identified manually.

3.2 The LP relaxation for the EFF for the Sparse TSP

From the single commodity flow formulation, we present its modification which we call the Embedded Flow Formulation (EFF) for the Sparse TSP. This formulation involves a polynomial number of constraints, even though the number of variables is increased considerably. The EFF for the Sparse TSP is given as:

$$\min_{x} \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} x_{ij} \qquad (1.7)$$

subject to

$$\sum_{j:(i<j)\in A} x_{ij} + \sum_{j:(j<i)\in A} x_{ji} = 2m_i, \quad \text{for all } i \in N \qquad (1.8)$$

$$y_{ij} - (n-1)x_{ij} \leq 0, \text{for all } (i, j) \in A \qquad (1.9)$$

$$\sum_{j:(i<j)\in A} y_{ij} - \sum_{j:(j<i)\in A} y_{ji} = -1, \quad \text{for all } i = 2, 3, \cdots, n \qquad (1.10)$$

$$\sum_{i \in A} (y_{1i} - y_{i1}) = n - 1 \qquad (1.11)$$

$$m_i \geq 0, \text{ for all } i \in N \qquad (1.12)$$

$$x_{ij} = 0 \text{ or } 1, \text{ for all } (i, j) \in A \qquad (1.13)$$

$$y_{ij} \geq 0, \text{ for all } (i, j) \in A \qquad (1.14)$$

The LP relaxation for the EFF for the Sparse TSP formulation is obtained by relaxing the integrality constraint (1.13). All other equations remain the same except constraint (1.13) changes to:

$$x_{ij} \geq 0, \text{ for all } (i, j) \in A \qquad (1.15)$$

It was interesting to know how close the lower bound Z_{LB} obtained by solving the subtour relaxation is to the length of an optimal tour Z_{opt}. Worst-case analysis of HK lower bound by Wolsey [9] and Shmoy and Williamson [10] show that for any cost function *C* satisfying the triangle inequality, the ratio Z_{LB}/Z_{opt} is at least $2/3$. The $2/3$ lower bound is not shown to be tight and actually it is conjectured by Goemans [11] that $(Z_{LB}/Z_{opt}) \geq 3/4$. Our computational results show that for many instances the above ratio is very close to 1.

The results we obtained are presented in figure 8. In general the LP relaxation is not equal to the minimum tour length but it is very close. LP relaxation for the ILP Sparse TSP gives a much tighter lower bound than the LP relaxation for the EFF for the Sparse TSP and requires fewer iterations.

3.3 An Arc-cutset Partial Enumeration Strategy (APES)

In this section we are proposing a strategy for quickly generating some of the violated arc-cutset constraints which we call an Arc-cutset Partial Enumeration Strategy (APES). The APES is based on the following observation, using the formulation for the ILP Sparse TSP, we can drop the connectivity constraints. When the resulting formulation is solved the solution produces a lot of disconnected components, most of which will have two nodes connected by two arcs. That is to say each component is a subtour, and we ended up having a lot of these subtours.

These components needed to be connected with other components to produce a single connected component. To be able to achieve this, we generated an arc-cutset constraint for each component. In other words we generated an arc-cutset

constraint for each arc in the graph. This approach is reasonable to the Sparse TSP because the number of arcs m in the sparse graph is $O(n)$ as opposed to $O(n^2)$ in the complete graph.

The arc-cutset constraints generated this way are all valid inequalities. Naddef and Rinaldi [12], Cornuéjols et al [13], and Swamy and Thulasiraman [14] have shown the validity of the arc-cutset constraints. They say that once the components are connected then the violated arc-cutset constraints are valid inequalities.
Our algorithm for the APES is given below.

An Arc-cutset Partial Enumeration Strategy algorithm
Step 1: Formulate the problem using evenness condition
Constraints and integrality constraints only.
Let *nodes* be the number nodes in the starting path
Let *nodesv* be the number of nodes to be visited
Let *k := 2*
Step 2: For *nodes* := *k* to *n* do
For *nodesv := 3* to *n – k* do
List all arcs incident to the path with end node 1 and *nodesv*
add the arc-cutset constraint to the formulation
EndFor
EndFor
Step 3: Solve the new formulation using any LP solver
Step 4: stop

In forming the arc-cutset constraints, we first used a subtour component consisting only of two end nodes *i* and *j* with *(i,j)* as a component. The violated arc-cutset constraints were constructed by listing all arcs incident to node *i* and node *j* to form one violated arc-cutset constraint, i.e., all arcs incident with a subtour component. The arc connecting node *i* and node *j* was not included in the arc-cutset constraints. Figure 2 shows how using component *(i,j)* arc-cutset constraints was formed. This is what takes place in step 2 of the APES algorithm.

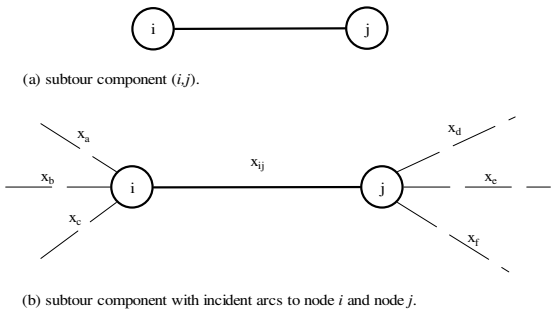

Figure 2: Formulation of the arc-cutset constraint

$$x_a + x_b + x_c + x_d + x_e + x_f \geq 2 \qquad (1.16)$$

The arc-cutset constraints which are generated by the APES are used to connect components. Since the APES starts by using the evenness condition constraints and integrality constraints, while omitting the connectivity constraints. For example the twenty nodes problem shown in figure 3, is used to demonstrate how the APES works.

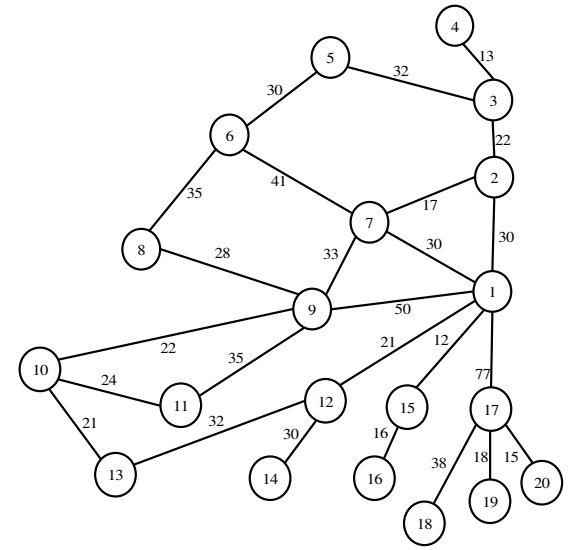

Figure 3: A twenty nodes problem to demonstrate how APES works

Solving the twenty nodes problems before adding the violated arc-cutset constraints generated by the APES gives the disconnected tours illustrated in figure 4 and whose objective function is 524.

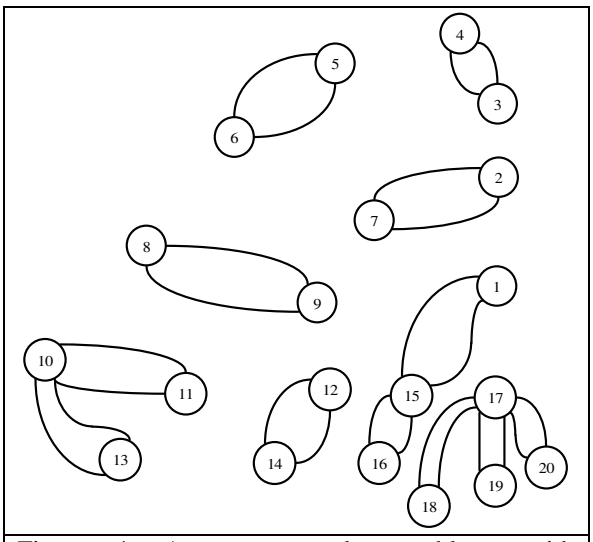

Figure 4: A twenty nodes problems with disconnected subtour components

After adding the violated arc-cutset constraints generate by the APES we got a tour illustrated in figure 5 and its objective function was 765, which in this case happened to be the optimal tour.

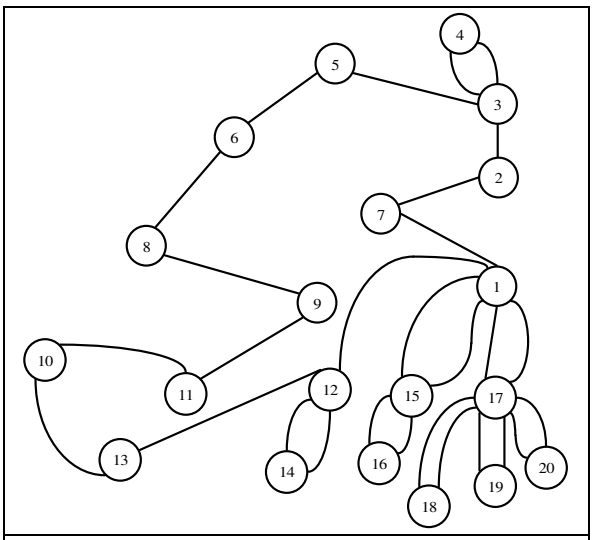

Figure 5: A twenty nodes problem tour

We carried out an Arc-cutset Partial Enumeration Strategy to generate violated arc-cutset constraints from each arc. After solving the problems we managed to get graphs with very few disconnected components. These needed a few more violated arc-cutset constraints to be identified for the graph to be connected. The APES was able to produce optimal tours for problems with up to nine nodes without adding any arc-cutset constraints. It is interesting to see how the APES we are proposing performed in some graphs. For example for the 30 nodes problem we had to add eleven more violated arc-cutset constraints before getting a tour, while we had to add only two more violated arc-cutset constraints for the 67 nodes problem to get a tour. Figure 8 shows the results we got from our test problems.

We then extended this technique of identifying violated arc-cutset constraints. These new violated arc-cutset constraints were formed by visiting a path of three or more nodes together. The first violated arc-cutset constraint was formed by visiting any three nodes which form a path. When forming these constraints, we included all arcs which were incident to nodes *1* or *2* or *3* and ignore arcs connecting nodes *1, 2,* and *3*. The next constraint was formed by adding the fourth node to the path and the violated arc-cutset constraint was identified by listing all the nodes incident to the path consisting of four nodes ignoring the arcs which form the path. The process continues until we had visited (*n - 2*) nodes in the graph. Figure 6 shows how these violated arc-cutset constraints were formed.

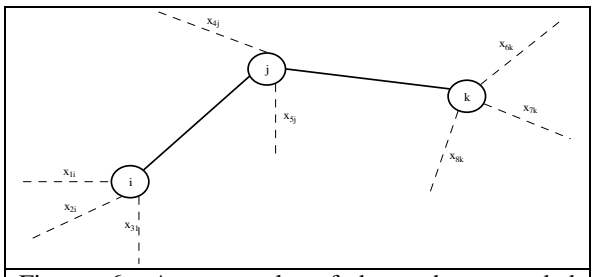

Figure 6: An example of how the extended technique for identifying violated arc-cutset constraints works

From figure 6 the following violated arc-cutset constraint (1.17) will be formed:

$$x_{1i} + x_{2i} + x_{3i} + x_{4j} + x_{5j} + x_{6k} + x_{7k} + x_{8k} \geq 2 \quad (1.17)$$

The results in figure XX shows how the APES method performed when the starting path visited 3, 4, … , 10 nodes together. In other words at first the starting path had 3 nodes and we extended the path by adding one node at a time. The second time the starting path had 4 nodes and we extended the path by adding one node at a time. We

continued increasing the number of nodes in a starting path, until at last our starting path had 10 nodes to start with. We got in a good number of cases substantial improvements in the lower bound as we increased the number of nodes in the starting path. However, as the number of nodes in the starting path went beyond 10 we got marginal improvement.

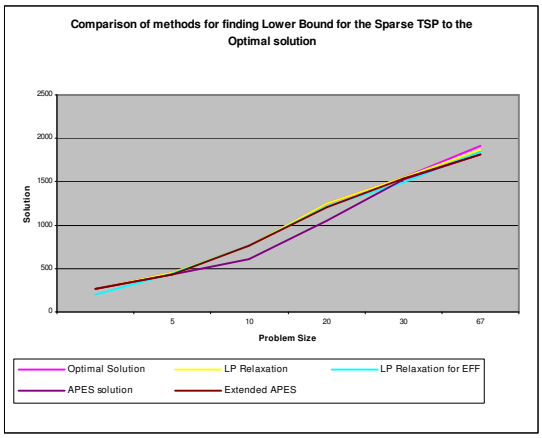

Figure 7: Comparison of methods for finding Lower Bound for the Sparse TSP to the optimal solution

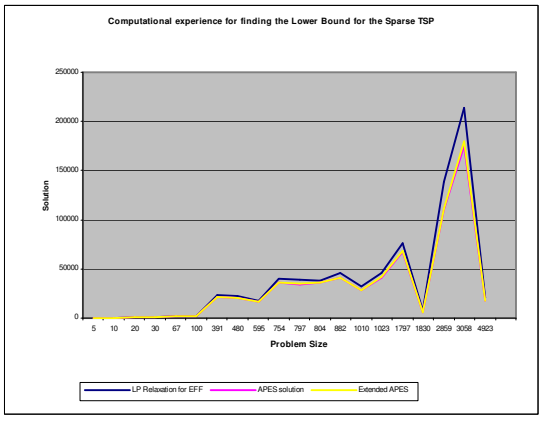

Figure 8: Computational experience for finding the Lower Bound for the Sparse TSP

The order of complexity of our APES is $O(n)$. This is far less than the connectivity constraints with the order of complexity of $O(2^n)$. This makes our strategy much easier to use.

4 Summary and future work

When optimization problems arise in practice we want to have confidence in the quality of the solutions. Quality guarantees are required because in most cases and especially for large problems, it is not always possible to find an optimal solution. Quality guarantees become possible by being able to compute good lower bounds at a reasonable computational cost. In this paper we have proposed a method for finding tight lower bound for the Sparse TSP using the LP relaxation method and the Arc-cutset Partial Enumeration Strategy.

When the LP relaxation method is used to find a lower bound for the ILP Sparse TSP, finding arc-cutset constraints is a headache especially for large problems. There are procedures for identifying violated arc-cutset constraints automatically in practice, such as the separation routines. These procedures are computational intensive and therefore were not used in this study.

The Arc-cutset Partial Enumeration Strategy proposed is a simple and fast way of getting a lower bound without spending time in a separation algorithm. However, computational results show that the lower bounds obtained by using this method are not very tight.

A lower bound on the optimal value (assuming a minimization problem) is obtained from a relaxation of the integer program. In the past ten to fifteen years attention has shifted from Lagrangian relaxation to Linear programming relaxation, since the latter type of relaxation can be strengthened more easily by using cutting planes. Combining cutting planes and Lagrangian relaxation usually causes convergence problems as discussed by Aardal et al [15]. LP relaxation gives the tightest lower bound of all lower bound techniques we have discussed in this paper.

References:

[1] Johnson, D.S., L.A. McGeoch, and E.E. Rothberg. *Asymptotic Experimental Analysis for the Held-Karp Traveling Salesman Bound*. in *Proceedings of the 7th Annual ACM-*

SIAM symposium on Discrete Algorithms. 1996.

[2] Reinelt, G., *The traveling salesman: Computational solutions for TSP applications*. 1994: Springer Verlag.

[3] Fleischmann, B., *A cutting plane procedure for travelling salesman problem on road networks.* European Journal of Operational Research, 1985. **21**: p. 307--317.

[4] Reinelt, G., *Fast Heuristics for Large Geometric Traveling Salesman Problems.* ORSA Journal on Computing, 1992. **4**(2): p. 206--217.

[5] Held, M. and R.M. Karp, *The Travelling Salesman Problem and Minimum spanning Trees, Part I.* Operations Research, 1970. **18**: p. 1138--1162.

[6] Dantzig, G., R. Fulkerson, and S. Johnson, *Solution of a large-scale Traveling-Salesman Problem.* Operations Research, 1954. **2**: p. 393--410.

[7] Valenzuela, C.L. and A.J. Jones, *Estimating the Held-Karp lower bound for the geometric TSP.* European Journal of Operational Research, 1996. **102**: p. 157--175.

[8] Held, M. and R.M. Karp, *The Travelling Salesman Problem and Minimum spanning Trees, Part II.* Mathematical Programming, 1971. **1**: p. 6--25.

[9] Wolsey, L.A., *Heuristic analysis, linear programming and Branch and Bound.* Mathematical Programming Study, 1980. **13**: p. 121--134.

[10] Shmoys, D.A. and D.P. Williamson, *Analyzing the Held-Karp TSP bound: A monotonicity property with application.* Information Processing Letters, 1990. **35**: p. 281--285.

[11] Goemans, M.X., *Worst-case Comparison of valid Inequalities for the TSP.* Mathematical Programming, 1995. **69**: p. 335--349.

[12] Naddef, D., *The binested inequalities for the symmetric traveling salesman Polytope.* Mathematics of Operations Research, 1992. **17**(4): p. 882--900.

[13] Cornuéjols, G., J. Fonlupt, and D. Naddef, *The travelling salesman problem on a graph and some related integer polyhedra.* Mathematical Programming, 1985. **33**: p. 1--27.

[14] Swamy, M.N.S. and K. Thulasiraman, *Graphs, Networks, and Algorithms*. 1981: John Wiley and Sons Ltd.

[15] Aardal, K., et al., *A decade of Combinatorial Optimization*. 1997, Working paper, Uetrecht University, Computer Science Department, Netherlands.