Performance of the Branch and Bound Algorithm on the Multistage Insertion Formulation of the Traveling Salesman Problem

Laleh Haerian Ardekani*, Tiru S. Arthanari*, Matthias Ehrgott†
*Department of Information Systems and Operations Management
†Department of Engineering Science
The University of Auckland
{l.haerian, t.arthanari, m.ehrgott}@auckland.ac.nz

Abstract

The traveling salesman problem (TSP) is probably the most studied difficult combinatorial optimization problem. The classical formulation for the TSP suggested by Dantzig, Fulkerson and Johnson (DFJ). The multistage insertion formulation (MI) is a compact formulation. Polytope given by the LP relaxation of the MI formulation for the TSP is proven to be as tight when projected into the DFJ variable space.

In this study we compare the performance of the branch and bound algorithm on the MI formulation for the TSP to other formulations. We define different branching rules using the MI formulation structure. We solve instances from the TSP library using the branch and bound algorithm on various TSP formulations. The MI formulation is found to perform better than other formulations in terms of solution time and the size of the branch and bound tree.

Key words: The Multistage Insertion Formulation, The Traveling Salesman Problem, Branch and Bound Method.

1 Introduction

The traveling salesman problem (TSP) was one of the first problems that was proven to be $\mathcal{NP}-$complete by Karp (1972) and has attracted a lot of attention from researchers. Given a complete graph $G = (V, E)$, where $V = \{1, .., n\}$ and $E = \{(i, j) \in V \times V | i \neq j\}$, the TSP is about finding the least cost Hamiltonian cycle in the graph.

Dantzig, Fulkerson and Johnson (1954) have suggested a formulation for the TSP (DFJ). Let the decision variables x_{ij}, for all nodes i and j in V be equal to one if the edge between i and j belongs to the solution, and be equal to zero otherwise. Let c_{ij} be the cost of edge $(i, j) \in E$. The DFJ formulation for the TSP is given by

constraints (1) to (3).

$$\min \sum c_{ij} x_{ij}$$

subject to:

$$\sum_{j<i} x_{ji} + \sum_{j>i} x_{ij} = \quad 2, \quad \forall i = 1, .., n, \tag{1}$$

$$\sum_{i,j \in S, i<j} x_{ij} \leq \quad |S| - 1, \quad \forall S \subseteq V, 2 \leq |S| \leq n-1, \tag{2}$$

$$x_{ij} \in \quad \{0,1\}, \quad \forall i, j \in V. \tag{3}$$

Constraints (1) guarantee that each node is connected to other nodes by exactly two edges. Constraints (2), also known as the *subtour elimination constraints*, make sure that there are no subtours (non-Hamiltonian cycles) in the solution. There are $O(2^{n-1})$ subtour elimination constraints in the DFJ formulation. The polytope given by the LP relaxation of the DFJ formulation is called *the subtour elimination polytope (SEP)*. The SEP is known to be a tighter polytope compared to the polytopes given by other TSP formulations, in the sense that it closely wraps around the TSP polytope.

The branch and bound method is based on the work by Dantzig, Fulkerson and Johnson (1959) on the TSP. The term *branch and bound* originates from the algorithm suggested by Little et. al. (1963). In a generic branch and bound method feasible solutions of an optimization problem are enumerated in order to find the optimal integer solution. The problem is constantly split into two subproblems by adding two mutually exclusive and exhaustive constraints (branching), and lower bounds are used to construct a proof of optimality without exhaustive search (bounding) (Papadimitriou and Steiglitz 1982). More studies on branch and bound for the TSP can be found in the works by Lawler and Wood (1966), Bellmore and Nemhauser(1968), and Balas and Toth (1985).

The multistage insertion formulation (MI) (Arthanari 1983) for the TSP is a compact formulation. It was shown that the polytope given by the LP relaxation of the MI formulation (P_{MI}) when projected into the variable space of the DFJ formulation, is a subset of the SEP (Arthanari and Usha 2000). In previous studies we compared the performance of the MI formulation for the TSP with other formulations (Haerian Ardekani, Arthanari, and Ehrgott 2010). We solved some problem instances from the TSP library, and some instances designed by Papadimitriou and Steiglitz (1978) for this purpose. The MI formulation is shown to outperform other formulations in terms of solution quality, solution time, and number of iterations, for both STSP and ATSP.

In this paper we compare the performance of the branch and bound method on various TSP formulations. The remainder of this paper is structured as follows. The MI formulation is given in Section 2. In Sections 3 and 4 we give a summary of various relaxations and branching rules used in the branch and bound method for the TSP. In Section 5 we suggest some branching rules based on the MI formulation. We report the computational results on applying branch and bound for various TSP

formulations on some TSP Library instances in Section 6. Section 7 includes the conclusions.

2 The Multistage Insertion (MI) Formulation for the STSP

The MI formulation for the STSP is based on constructing STSP tours by sequentially inserting nodes into the initial tour of three nodes 1, 2 and 3. Given a set of nodes $V = \{1, ..., n\}$, nodes from 4 to n are inserted sequentially between the nodes of this tour. Let x_{ijk} be equal to one if node k is inserted between nodes i and j, for $1 \leq i < j \leq k-1$ and $4 \leq k \leq n$, and be zero otherwise. Let c_{ij} be the cost of an edge $(i, j) \in E_n = \{(u, v) | 1 \leq u < v \leq n\}$, and Let $C_{ijk} = c_{ik} + c_{jk} - c_{ij}$. The MI formulation (Arthanari 1983) is:

$$\min \sum_{k=4}^{n} \sum_{(1 \leq i < j \leq k-1)} C_{ijk} x_{ijk}$$

subject to:

$$\sum_{1 \leq i < j \leq k-1} x_{ijk} = \quad 1, \quad 4 \leq k \leq n, \tag{4}$$

$$\sum_{k=4}^{n} x_{ijk} \leq \quad 1, \quad 1 \leq i < j \leq 3, \tag{5}$$

$$-\sum_{r=1}^{i-1} x_{rij} - \sum_{s=i+1}^{j-1} x_{isj} + \sum_{k=j+1}^{n} x_{ijk} \leq \quad 0, \quad 1 \leq i < j, 4 \leq j \leq n-1, \tag{6}$$

$$x_{ijk} \in \{0, 1\}, \quad 1 \leq i < j \leq k-1, 4 \leq k \leq n. \tag{7}$$

Constraints (4) of the formulation guarantees that each node from 4 to n is inserted in an edge. Constraint (5) ensures that at most one node is inserted in each of the edges of T_3. Constraint (6) makes sure that a node is inserted into an edge of the subtour only if that edge has been generated by previous insertions and is available. The MI formulation has $O(n^3)$ variables and $O(n^2)$ constraints.

3 Relaxations for the TSP

In order for the branch and bound method to perform well, it is important to start with an LP relaxation of the IP problem with a small gap. Different relaxations of the TSP have been considered by researchers to apply in the branch and bound methods. We give some of these results below.

The assignment problem (AP) relaxation of the ATSP is used by Eastman (1958), Little et al. (1963), and Bellmore and Maloneyear(Bellmore and Malone 1971). The AP is given by the objective function of the DFJ formulation, and constraints (8)– (11).

$$\sum_{(i,j) \in A} x_{ij} = \quad 1, \quad \forall i \in V, \tag{8}$$

$$\sum_{(i,j)\in A} x_{ij} = 1, \quad \forall j \in V, \tag{9}$$

$$x_{ij} \leq 1, \quad \forall i,j \in V, \tag{10}$$

$$x_{ij} \geq 0, \quad \forall i,j \in V. \tag{11}$$

The solution to the AP is either a directed tour or a set of directed subtours. Eastman used the network flow algorithm by Ford and Fulkerson (1956) to solve the AP. The AP can be solved using the Hungarian method in $O(n^3)$ time (Ahuja, Magnanti, and Orlin 1993).

The 2-matching relaxation is used by Bellmore and Malone (1971) for the STSP. The answer to the 2-matching problem is either a tour or a collection of subtours. The 2-matching problem has the same objective function as the DFJ formulation subject to constraints (1), (10), and (11).

The 1-tree relaxation of the STSP is first used by Held and Karp (1971) and Christofides (1970). Let V' be $V - \{1\}$. This relaxation is given by constraints (10), (11), and constraints (12) – (14).

$$\sum_{(i,j)\in S\times(V'-S), j>i} x_{ij} + \sum_{(i,j)\in(V'-S)\times S, j>i} x_{ij} \geq 1, \quad \forall S \subset V', \tag{12}$$

$$\sum_{i\in V}\sum_{j>i} x_{ij} = n, \tag{13}$$

$$\sum_{i\in V} x_{1j} = 2, \tag{14}$$

The n-path problem is about finding the shortest path in a graph that includes n nodes (n-path) starting and ending at some node $v \in V$. The LP relaxation of the $n-$path problem is first used by Houck et al. (1980) for the TSP. The $n-$path problem can be solved using dynamic programming in $O(n^3)$ steps (Balas and Toth 1985).

The LP with cutting planes was first suggested by Gomory (1958) for solving generic integer programming optimization problems. Crowder and Padberg (1980) used this solution method for the STSP. The main feature of their method is finding appropriate inequalities to use as cutting planes in each step (Balas and Toth 1985).

4 Branching Methods

We borrow the notations used by Balas and Toth (1985), for defining the subproblems in a branching tree. Starting with the root problem labeled as problem 1, we use string labels for the subproblems in a way that they show the hierarchy of the problem in the branch and bound tree. Given some problem m in the branching tree, let $\mathcal{E}_m$ indicate the set of edges (i,j) that are excluded from problem m, and let $\mathcal{I}_m$ indicate the set of edges included in the problem. Using variables x_{ij}, the sets $\mathcal{E}_m$ and $\mathcal{I}_m$ for some subproblem m can be defined by the following condition.

$$\begin{cases} (i,j) \in \mathcal{I}_m, & \text{if } x_{ij} = 1 \text{ in problem } m, \\ (i,j) \in \mathcal{E}_m, & \text{if } x_{ij} = 0 \text{ in problem } m. \end{cases} \tag{15}$$

The solution to problem m is given subject to the constraints of the predecessor problem of m and subject to condition (15). Given problem m, let the r^{th} successor of problem m be labeled as problem mr. We use the notations $\mathcal{I}_{mr}$ and $\mathcal{E}_{mr}$ to indicate the set of variables that are included in and excluded from mr problem respectively.

Many different branching rules have been suggested for the DFJ formulation. Little et al. (1963) designed a branching rule that creates two subproblems where some arc (i, j) is included in one branch and excluded from the other branch. This rule does not use the structure in the TSP and can be applied to any IP problem (Balas and Toth 1985). Little et al. do not solve the LP at each node but they calculate a lower bound for the LP using the dual solution. Their algorithm has more branching steps but requires less computation at each step compared to other methods (Lawler and Wood 1966). At each branching step, Little et al. (1963) select a pair (i, j) such that the subproblem corresponding to $x_{ij} = 0$ would give a bound as large as possible (Lawler and Wood 1966). The branching rule can be expressed as follows.

$$\begin{cases} \mathcal{E}_{m1} = \mathcal{E}_m \cup \{(i,j)\}, & \mathcal{I}_{m1} = \mathcal{I}_m, \\ \mathcal{E}_{m2} = \mathcal{E}_m, & \mathcal{I}_{m2} = \mathcal{I}_m \cup \{(i,j)\}. \end{cases} \tag{16}$$

For more branching rules and TSP relaxations we refer the reader to the works by Balas and Toth (1985), and Applegate et al. (2006).

5 Branch and Bound Method for the MI Formulation

Similar to the notations used in the previous section for x_{ij} variables, we use the following notations based on the MI formulation structure to define subproblems in a branch and bound tree. Let the sets $\mathcal{E}_m$ and $\mathcal{I}_m$ correspond to the sets of x_{ijk} variables that are excluded from or included in the solution of some subproblem m respectively. We define $\mathcal{E}_m$ and $\mathcal{I}_m$ by the following condition for the subproblem m.

$$\begin{cases} (i,j,k) \in \mathcal{I}_m, & \text{if} x_{ijk} = 1, \\ (i,j,k) \in \mathcal{E}_m, & \text{if} x_{ijk} = 0. \end{cases} \tag{17}$$

We can use sets $\mathcal{E}_m$ and $\mathcal{I}_m$ to define subproblems $m1$ and $m2$ that partition the solution space of problem m into two regions.

Example 1 Consider this solution to a 13-city MI-relaxation problem, $x_{2,3,4} = 1$, $x_{1,2,5} = 1$, $x_{1,3,6} = 1$, $x_{2,5,7} = 1/2$, $x_{2,3,4} = 1$, $x_{2,3,4} = 1$, $x_{2,3,4} = 1$, $x_{2,3,4} = 1$, $x_{2,3,4} = 1$, $x_{2,3,4} = 1$, $x_{2,3,4} = 1$, $x_{3,6,7} = 1/2$, $x_{3,7,8} = 1/2$, $x_{6,7,8} = 1/2$, $x_{2,4,9} = 1$, $x_{3,6,10} = 1/2$, $x_{6,8,10} = 1/2$, $x_{7,8,11} = 1$, $x_{3,8,12} = 1/2$, $x_{3,10,12} = 1/2$, and $x_{6,10,13} = 1$. Having $x_{3,8,12} = 0.5$, and $x_{3,10,12} = 0.5$, suggests that edge $(3, 12)$ is created by having variables $x_{3,j,12}$ to take values greater than zero. We use the following branching rule that partitions the solution space into two regions $\sum_{i\neq 3, i<j<12} x_{i,j,12}$, and $\sum_{i=3, i<j<12} x_{i,j,12}$. Similarly for $k = 8$ we have $x_{3,7,8} = 0.5$, and $x_{6,7,8} = 0.5$, so we can have $\sum_{j\neq 7} x_{ij8} = 0$, and $\sum_{j=7} x_{ij8} = 0$. And finally based on having $x_{3,6,10} = 0.5$ and $x_{6,8,10} = 0.5$, we can use $\sum_{i\neq 6 \wedge j\neq 6} x_{i,j,10} = 0$, and $\sum_{i=6 \vee j=6} x_{i,j,10} = 0$, to partition the solution space into two regions.

The following branching rules generalize the branching formulations given in Example 1. These rules partition the solution space into at least two regions

MI Branching Rule 1 (MIR_1)

Given the solution to some problem m, if for some $\hat{k}$ and some $\hat{i}$ we have $0 < x_{\hat{i}j\hat{k}} < 1$, then the successors of m are partitioned into two groups based on the following rules.

$$\begin{cases} \mathcal{E}_{m1} = \mathcal{E}_m \cup \{(i,j,k) | i = \hat{i}, k = \hat{k}, i < j < k\}, \mathcal{I}_{m1} = \mathcal{I}_m, \\ \mathcal{E}_{m2} = \mathcal{E}_m, I_{m2} = \mathcal{I}_m \cup \{(i,j,k) | i = \hat{i}, k = \hat{k}, i < j < k\}, \end{cases} \tag{18}$$

Similarly if for some $\hat{k}$ and some $\hat{j}$ for some problem m we have $0 < x_{i\hat{j}\hat{k}} < 1$, then the successors of m are partitioned into two groups using the following rules.

$$\begin{cases} \mathcal{E}_{m1} = \mathcal{E}_m \cup \{(i,j,k) | j = \hat{j}, k = \hat{k}, 1 < i < j\}, \mathcal{I}_{m1} = \mathcal{I}_m, \\ \mathcal{E}_{m2} = \mathcal{E}_m, \mathcal{I}_{m2} = \mathcal{I}_m \cup \{(i,j,k) | j = \hat{j}, k = \hat{k}, 1 < i < j\}, \end{cases} \tag{19}$$

MI Branching Rule 2 (MIR_2)

Given some problem m, if for some $\hat{k}$ and some $\hat{i}$, we have $0 < x_{\hat{i_1}\hat{j}k}, x_{\hat{i_2}\hat{j}k} < 1$, the successors of m can be defined as follows.

$$\begin{cases} \mathcal{E}_{m1} = \mathcal{E}_m, \mathcal{I}_{m1} = \mathcal{I}_m \cup \{(\hat{i_1}\hat{j}\hat{k})\}. \\ \mathcal{E}_{m2} = \mathcal{E}_m, \mathcal{I}_{m2} = \mathcal{I}_m \cup \{(\hat{i_2}\hat{j}\hat{k})\}. \\ \mathcal{E}_{m3} = \mathcal{E}_m \cup \{(\hat{i_1}\hat{j}\hat{k}), (\hat{i_2}\hat{j}\hat{k})\}, \mathcal{I}_{m3} = \mathcal{I}_m. \end{cases} \tag{20}$$

Similarly, for some $\hat{k}$ and some $\hat{i}$, we have $0 < x_{\hat{i}\hat{j_1}k} < 1$, and $0 < x_{\hat{i}\hat{j_2}k} < 1$, the successors of m can be defined as follows.

$$\begin{cases} \mathcal{E}_{m1} = \mathcal{E}_m, \mathcal{I}_{m1} = \mathcal{I}_m \cup \{(\hat{i}\hat{j_1}\hat{k})\} \\ \mathcal{E}_{m2} = \mathcal{E}_m, \mathcal{I}_{m2} = \mathcal{I}_m \cup \{(\hat{i}\hat{j_2}\hat{k})\} \\ \mathcal{E}_{m3} = \mathcal{E}_m \cup \{(\hat{i}\hat{j_1}\hat{k}), (\hat{i}\hat{j_2}\hat{k})\}, \mathcal{I}_{m3} = \mathcal{I}_m. \end{cases} \tag{21}$$

MI Branching Rule 3 (MIR_3)

This branching rule is the same as the generic branching rule for the IP methods. Given an MI relaxation solution for some $0 < x_{\hat{i}\hat{j}\hat{k}} < 1$, we define the following branching rule.

$$\begin{cases} \mathcal{E}_{m1} = \mathcal{E}_m, \mathcal{I}_{m1} = \mathcal{I}_m \cup \{(\hat{i}, \hat{j}, \hat{k})\}, \\ \mathcal{E}_{m2} = \mathcal{E}_m \cup \{(\hat{i}, \hat{j}, \hat{k})\}, \mathcal{I}_{m2} = \mathcal{I}_m, \end{cases} \tag{22}$$

When choosing x_{ijk} variables to branch on, we have the option of choosing variables with k values as close to n or as close to 4 as possible. For example in the solution given in Example 1, we can choose between branching on x_{ijk} variables with $k = 7$ or $k = 12$. We represent the combination of a branching rule MIR_i with branching on variables of either greatest value of k or smallest values of k, using $\text{MIR}_{i,1}$, and $\text{MIR}_{i,2}$, respectively.

In the next section we give some computational results on these different branching rules for some TSPLIB (Reinelt 1995) instances.

Table 1: CPU Second for Branch and Bound Method with MI Branching Rules

Problem	$MIR_{1,1}$	$MIR_{2,1}$	$MIR_{3,1}$	$MIR_{1,2}$	$MIR_{2,2}$	$MIR_{3,2}$
bayg29	1.13	1.41	1.04	**0.75**	1.97	1.97
bays29	1.47	1.43	2.23	**1.00**	3.23	3.29
dantzig42	5.44	3.27	6.07	**2.34**	3.38	5.22
swiss42	**1.42**	1.83	1.45	1.66	3.48	2.40
att48	**4.70**	8.69	8.46	16.89	24.52	17.18
hk48	2.93	3.64	**2.92**	9.31	12.70	12.53
brazil58	**6.23**	8.34	6.98	6.65	9.09	6.77
st70	408.91	588.33	629.31	98.46	**59.38**	206.07
eil76	187.67	504.81	**139.58**	355.28	408.42	851.87
rd100	**305.99**	441.49	361.57	394.76	652.46	444.35
eil101	**1209.30**	3342.42	3956.10	2254.70	3618.60	3693.50
lin105	**113.78**	148.27	119.61	187.84	538.37	384.00
gr120	$> 10^4$	$> 10^4$	$> 10^4$	-	-	-

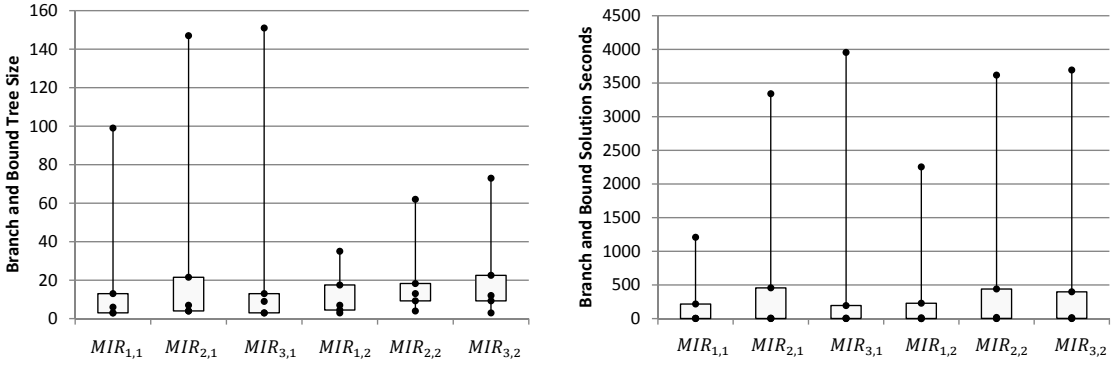

Figure 1: Branch and Bound Solution Seconds (left) and Tree Size (right) for MI Branching Rules

6 Computational Results on Branch and Bound for Various TSP Formulations

We applied the different MI branching rules, $\text{MIR}_{i,j}$ for $i = 1, 2, 3$ and $j = 1, 2$, on some TSPLIB instances. We use Cplex 9.1 for solving the subproblems in each tree node. The solution times are compared in Table 1 and the sizes of the branch and bound trees are compared in Table 2. These results are also illustrated in Figure 1. The median, minimum and maximum values of the solution times and the sizes of the trees are shown in this figure. The first and third quartiles are illustrated using boxes. From Figure 1 we can observe that $\text{MIR}_{1,1}$ and $\text{MIR}_{1,2}$ provide smaller branch and bound trees compared to other methods. The solution times for $\text{MIR}_{1,1}$ are less than those for other rules, except for problem st70, a Euclidean 70-city problem. For the MIR_1 rules, branching on the variables with smallest value of k seems to perform better than using larger values for k.

We applied branch and bound with the DFJ (1954), Wong (1980), Claus (1984), and Carr (1996) formulations and compared with branch and bound on the MI formulation. The results are given in Table 2 and 3, and illustrated in Figure 2. The number of violated subtour elimination constraints for the DFJ formulation that are found and added to the subproblems are shown in Table 3 in column *SEC*. Apart from the DFJ formulation, all the three MI branching rules provide the smallest size

Table 2: Size of the Branch and Bound Tree

Problem	Carr	Claus	DFJ	$MIR_{1,1}$	$MIR_{2,1}$	$MIR_{3,1}$	$MIR_{1,2}$	$MIR_{2,2}$	$MIR_{3,2}$	Wong
bayg29	7	19	5	5	7	5	5	**3**	10	19
bays29	11	33	7	7	7	11	11	**5**	16	33
dantzig42	11	55	5	11	7	13	13	5	7	49
swiss42	5	17	5	**3**	4	**3**	**3**	7	5	27
att48	9	25	7	**5**	10	9	17	25	17	27
hk48	7	7	7	**3**	4	**3**	9	13	13	13
brazil58	3	25	7	**3**	4	**3**	**3**	4	**3**	21
st70	-	-	117	99	147	151	19	**13**	39	-
eil76	-	-	11	19	53	**13**	33	41	73	-
rd100	-	-	7	**7**	11	9	9	13	9	-
eil101	-	-	47	**19**	70	81	35	62	63	-
lin105	-	-	**3**	**3**	4	**3**	5	13	9	-
gr120	-	-	2055	**71**	90	93	-	-	-	-

Table 3: Solution Seconds for Branch and Bound on Different Methods

Problem	Carr	Claus	DFJ	SEC	$MIR_{1,1}$	$MIR_{2,1}$	$MIR_{3,1}$	Wong
bayg29	29.87	136.07	3.97	65	1.15	1.41	**1.04**	558.0
bays29	45.88	218.88	2.26	34	1.47	**1.43**	2.23	934.7
dantzig42	547.43	4706.28	3.37	32	5.44	**3.27**	6.07	13777.5
swiss42	260.33	1418.20	2.85	44	**1.42**	1.83	1.45	4878.5
att48	3097.80	6765.88	9.26	285	**4.70**	8.69	8.46	15277.8
hk48	1964.80	2415.82	4.84	139	2.93	3.64	**2.92**	6764.8
brazil58	5075.50	18168.93	6.53	131	**6.23**	8.34	6.98	23619.5
st70	-	-	**125.21**	90757	408.91	588.33	629.31	-
eil76	-	-	**12.70**	106	187.67	504.81	139.58	-
rd100	-	-	**43.15**	456	305.99	441.49	361.57	-
eil101	-	-	**174.90**	14006	1209.30	3342.42	3956.10	-
lin105	-	-	**46.17**	138	113.78	148.27	119.61	-
gr120	-	-	**5411.50**	4930140200	$> 10^4$	$> 10^4$	$> 10^4$	-

for branching trees and require the least amount of computational time. For the DFJ formulation, the size of the branch and bound tree is greater than MI, except for problem eil76, but the computational time for the DFJ formulation is less than for the MI formulation. This is probably due to the small size of the LPs solved at each branch and bound node. The size of the branch and bound tree is significantly larger for the DFJ formulation for problem gr120 which is a 120-city problem with geographical distances compared to the MI formulation. The MI branching methods provide smaller branch and bound trees for gr120, although their solution time is greater than that of the DFJ formulation.

7 Conclusion

In this paper we suggested three branching rules for the MI formulation of the TSP, we refer to as MIR_1, MIR_2, and MIR_3. We applied these three branching rules in branch and bound algorithms and found rule MIR_1 to perform better than the other two rules. We also compared the performance of branch and bound method on various TSP formulations by solving some TSPLIB instances. We found that the branch and bound method on the MI formulation performed better than other TSP

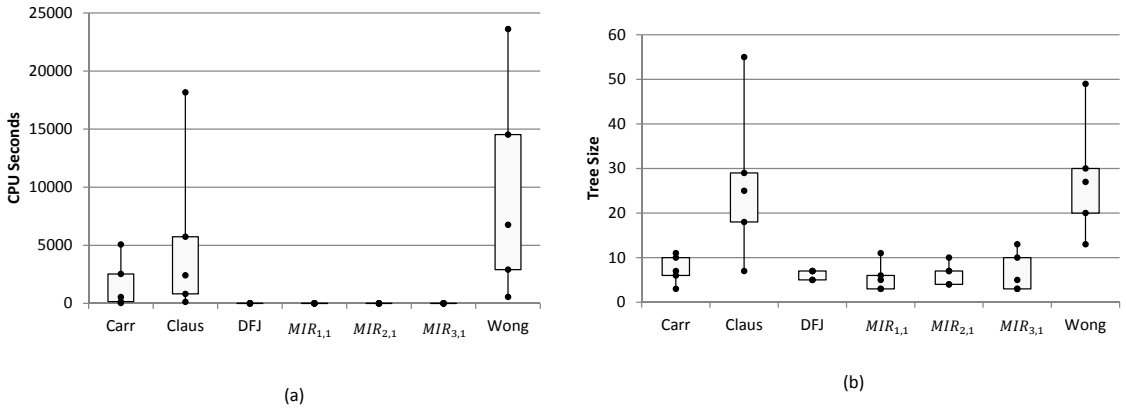

Figure 2: Branch and Bound Solution Seconds (a) and Tree Size (b) for TSP formulations

formulations in terms of solution time and the size of the branch and bound tree. Considering the small solution time for instances smaller than 58, it seems feasible to use the branch and bound method with the branching rule $\text{MIR}_{1,1}$ for solving such TSP instances.

References

Ahuja, R.K., T.L. Magnanti, and J.B. Orlin. 1993. *Network Flows.* Prentice-Hall, New Jersey.

Applegate, D.L., R.E. Bixby, V. Chvtal, and W.J. Cook. 2006. *The Traveling Salesman Problem: A Computational Study.* Princton University Press, New Jersey.

Arthanari, T.S. 1983. "On the traveling salesman problem." Edited by A. Bachem and et. al., *Mathematical Programming - The State of the Art.* Springer-Verlag.

Arthanari, T.S., and M. Usha. 2000. "An alternate formulation of the symmetric traveling salesman problem and its properties." *Discrete Applied Mathematics* 98 (3): 173–190.

Balas, E., and P. Toth. 1985. "Branch and Bound Methods." In *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, edited by E.L. Lawler, J.K. Lenstra, A.H.G. Rinooy Kan, and D.B. Shmoys, 361–402. Wiley, New York.

Bellmore, M., and J.C. Malone. 1971. "Pathology of traveling-salesman subtour-elimination algorithms." *Operations Research* 19 (2): 278–307.

Bellmore, M., and G.L. Nemhauser. 1968. "The traveling salesman problem: A survey." *Operations Research* 16 (3): 538–558.

Carr, R. 1996. "Separating over classes of TSP inequalities defined by 0 node-lifting in polynomial time." In *Integer Programming and Combinatorial Optimization*, edited by E. Lawler, J.K. Lenstra, A.H.G. Rinooy Kan, and D.B.S Shmoys, 460–474. Springer Berlin Heidelberg.

Christofides, N. 1970. "The shortest Hamiltonian chain of a graph." *SIAM Journal on Applied Mathematics* 19 (4): 689–696.

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

Crowder, H., and M.W. Padberg. 1980. "Solving large-scale symmetric travelling salesman problems to optimality." *Management Science* 26 (5): 495–509.

Dantzig, G.B., D.R. Fulkerson, and S.M. Johnson. 1954. "Solution of a large-scale traveling-salesman problem." *Operations Research* 2 (4): 393–410.

———. 1959. "On a linear-programming, combinatorial approach to the traveling-salesman problem." *Operations Research* 7 (1): 58–66.

Eastman, W.L. 1958. "Linear Programming with Pattern Constraints." Ph.D. diss., Harvard University, Boston, MA.

Ford, L.R., and D.R. Fulkerson. 1956. "Solving the transportation problem." *Management Science* 3 (1): 24–32.

Gomory, R.E. 1958. "Outline of an algorithm for integer solutions to linear programs." *Bulletin of the American Mathematical Society* 64 (5): 275–278.

Haerian Ardekani, L., T.S. Arthanari, and M. Ehrgott. 2010. "The multistage insertion formulation of the symmetric traveling salesman problem - An empirical study." Department of Information Systems and Operations Management, Business School, The University of Auckland/Department of Engineering Science, Faculty of Engineering, The University of Auckland. Working Paper No. 319, 18pp.

Held, M., and R.M. Karp. 1971. "The traveling-salesman problem and minimum spanning trees: Part II." *Mathematical Programming* 1 (1): 6–25.

Houck Jr., D.J., J.C. Picard, M. Queyranne, and R.R. Vemuganti. 1980. "The traveling salesman problem as a constrained shortest path problem: Theory and computational experience." *Opsearch* 17:93–109.

Karp, R.M. 1972. "Reducibility Among Combinatorial Problems." In *Complexity of Computer Computations*, edited by R.E. Miller and J.W. Thatcher, 85–103. Plenum New York.

Lawler, E.L., and D.E. Wood. 1966. "Branch-and-bound methods: A survey." *Operations Research* 14 (4): 699–719.

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

Papadimitriou, C.H., and K. Steiglitz. 1978. "Some examples of difficult traveling salesman problems." *Operations Research* 26 (3): 434–443.

———. 1982. *Combinatorial Optimization Algorithms and Complexity*. Printce-Hall.

Reinelt, G. 1995. TSPLIB. http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95.

Wong, R.T. 1980. "Integer programming formulations of the travelling salesman problem." *IEEE Conf. on Circuits and Computers*. 149–152.