Fall 8-26-2019

An Empirical Study of Mixed Integer Programming Formulations of the Backhaul Profit Maximization Problem

Yulan Bai
yulanb@smu.edu

Eli V. Olinick
olinick@smu.edu

An Empirical Study of Mixed Integer Programming Formulations of the Backhaul Profit Maximization Problem

Yulan Bai and Eli Olinick

Department of Engineering Management, Information and Systems,
Southern Methodist University, United States

ABSTRACT

Solving an instance of the Backhaul Profit Maximization Problem (BPMP) requires simultaneously solving two problems: (1) determining how to route an empty delivery vehicle back from its current location to its depot by a scheduled arrival time, and (2) selecting a profit-maximizing subset of delivery requests between various locations on the route subject to the vehicle's capacity. We propose and test a series of enhancements to the node-arc and triples mixed integer programming formulations of BPMP found in the literature and develop a multi-criteria Composite Index Method (CIM) to evaluate the results. We find that CPLEX takes 5 to 34 minutes (real time) to solve BPMP instances from the literature with 20 potential pickup/drop-off locations using the original node-arc model on the computers in our lab. Applying our own insights and adapting techniques from the literature on related problems, we develop an enhanced node-arc formulation that reduces the range of solution times of the 20-location instances to 31 to 105 seconds. Additionally, we solve problem instances that are twice as large (40 locations) as those solved in the literature. With the triples formulation from the literature, we find that the 20-location instances take between 2 to 21 seconds to solve. Using our enhanced triples formulation, however, we solve these same instances in six seconds or less. Comparing our two enhanced formulations, we solve 40-location instances in an average of 7 minutes with the enhanced triples formulation compared to an average of 92 hours with the enhanced node-arc formulation. Additionally, we find that the average time to solve 50-node instances with the enhanced triples formulation is 36 minutes.

Keywords: Freight Logistics, Pickup and Delivery, Backhauls, Mixed Integer Programming, Multicommodity Flows, Composite Index Method (CIM)

1. INTRODUCTION

The Backhaul Profit Maximization Problem (BPMP) requires simultaneously solving two problems: (1) determining how to route an empty delivery vehicle back from its current location to its depot by a scheduled arrival time, and (2) selecting a profit-maximizing subset of delivery requests between various locations on the route subject to the vehicle's capacity. Figure 1 illustrates a BPMP instance and solution.

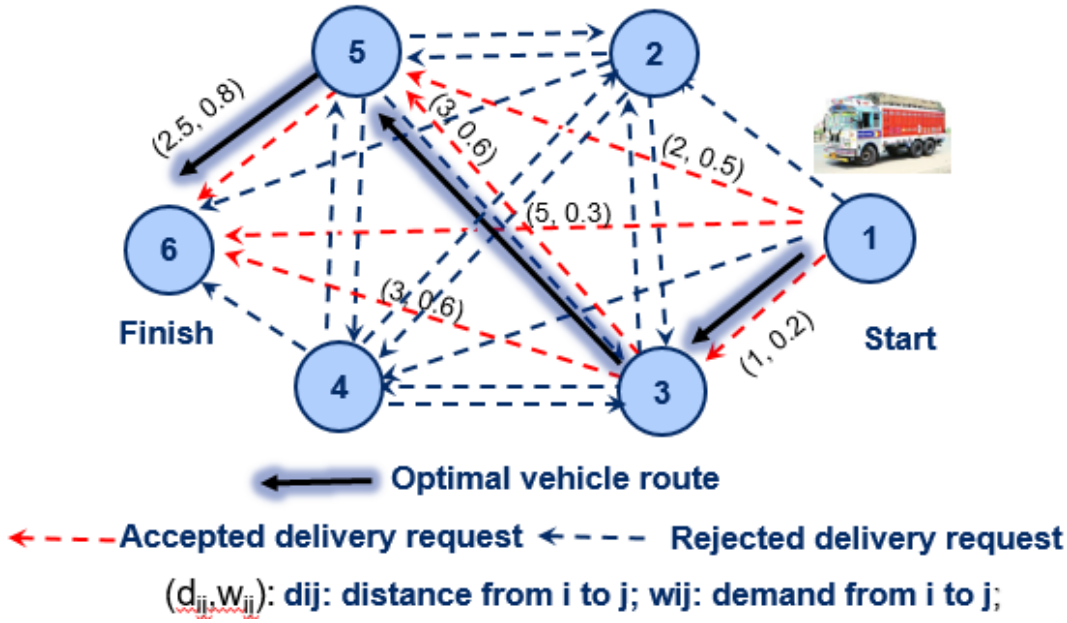

Figure 1. BPMP example.

Figure 1 shows a network representation of the problem with an empty vehicle at a location represented by node 1. The vehicle weighs 1 ton and has a carrying capacity of $Q = 2$ tons of cargo. The vehicle needs to return to its depot, represented by node 6, within a fixed period of time. The vehicle's average traveling speed limits the route to node 6 to a maximum distance of 7 miles. The vehicle can make extra money by accepting delivery requests to pick up cargo at the locations represented by nodes 1 through 5, destined for locations represented by nodes 2 through 6 as long as it can get back to the depot on time. The tuple (d_{ij}, w_{ij}) indicates the distance (in miles) and the size of the delivery request (in tons) from node i to node j. The solution indicated in Figure 1 routes the vehicle on the path represented by the arc sequence (1, 3), (3, 5), (5, 6). The dashed, red arcs in the figure indicate that vehicle makes the following pickups and dropoffs:

- Node 1: pick up 0.2 tons destined for node 3, 0.5 tons destined for node 5, and 0.3 tons destined for node 6. Carry one ton of cargo to one mile to Node 3.
- Node 3: drop off 0.2 tons from node 1; pick up 0.6 tons destined for node 5 and another 0.6 tons destined for node 6. Carry two tons of cargo three miles to Node 5.
- Node 5: drop off 0.5 and 0.6 tons from nodes 1 and 3, respectively; Pick up 0.8 tons destined for node 6. Carry 1.7 tons of cargo 2.5 miles to Node 6.
- Node 6: drop off 0.3, 0.6, and 0.8 tons from nodes 1, 3, and 5, respectively.

The net profit for the solution indicated in Figure 1 is the revenue generated from the accepted delivery requests minus the transportation costs. In the literature, the revenue for delivering w_{ij} truckloads from location i to location j is assumed to be proportional to the direct distance, d_{ij}, and the transportation cost is assumed to be function of the total distance traveled (6.5 miles in the Figure 1 example), and the total ton-miles carried (e.g., 11.25 ton-miles in Figure 1). The time constraint is treated as a distance constraint by assuming a given average driving speed for the vehicle, in our example the limit is a maximum distance of 7 miles.

To the best of our knowledge, BPMP was first introduced by Dong et al. (2006) who presented a heuristic for a special case where $w_{ij} = Q$, all delivery requests. Yu and Dong (2013) considered the general case in which $w_{ij} \leq Q$; that isdelivery requests are allowed to be equal to or less than the vehicle capacity (less than truck load, LTL). They proposed a genetic algorithm and a mixed integer programming formulation based on the traditional node-arc model of multicommodity flow that we refer to as the *node-arc formulation* of BPMP. In her dissertation, Dong (2015) proposed an alternative mixed integer programming formulation of BPMP called the *triples formulation*based on a compact formulation of multicommodity flow originally proposed by Matula (1986) for the maximum concurrent flow problem. Thus, there are two kinds of BPMP mixed integer programming formulations (also called models, used interchangeably hereinafter): node-arc and triples. Dong (2015) showed that the triples formulation has a significantly smaller constraint matrix and stronger linear programming (LP) relaxation than the node-arc formulation, and presented computational results in which CPLEX solved problem instances with up to 20 locations 90 to 2,000 times faster with the triples formulation. Dong (2015) was unable to solve larger problems with the node-arc formulation, but solved problems with up to 40 locations in an average of 90 minutes of CPU time using the triples formulation.

In this study we enhance both models by adapting techniques from the literature on related problems and applying our own insights into BPMP, and present results from an extensive empirical study in order to make a more comprehensive comparison of the two models and strengthen the case for the triples model. We review the models in Section 2 and describe the design and analysis of our experiments in the Section 3. Sections 4 and 5 describe the development of our enhanced node-arc and triples formulations, respectively. In Section 6 we compare the performance of CPLEX with the two enhanced formulations.

Our study makes the following contributions to the BPMP literature:

1. We solve larger problems than previously solved in the literature with our enhanced formulations: 40 vs. 20 locations and 50 vs. 40 locations for the node-arc and triples models, respectively.
2. We show that our enhanced models require significantly less time to solve than the original models proposed in the literature.
3. We develop a multi-criteria Composite Index Method (CIM) to compare the effectiveness of two models for the same problem.
4. We strengthen the case made by Dong (2015) and Dong et al. (2015) for using the triples representation of multicommodity flow in other appropriate applications besides BPMP.

2. ORIGINAL FORMULATIONS OF BPMP

The following presentation of the node-arc and triples formulations is adapted from Yu and Dong (2013) and Dong (2015).

2.1 Notation Common to Both Models

Parameters

V A set of locations (nodes) including the origin (1) and destination (n), $\{1,2, \cdots, n\}$
A A set of arcs, $\{(i, j): i \neq j,\ i \in V\backslash\{n\},\ j \in V\backslash\{1\}\}$
p Unit price charged to accept delivery request, dollars/mile/ton
c Unit travel cost incurred, dollars/mile/ton
Q The capacity of the vehicle, tons
v The weight of the vehicle, tons
D The maximum distance the vehicle can travel, miles
w_{kl} The weight of a customer's delivery request from k to l, tons
d_{kl} The Euclidean distance from k to l, miles

Common decision variables for both models

$x_{ij} \in \{0,1\}$: 1, if the vehicle travels directly from i to j on arc $(i,\ j) \in A$; 0 otherwise
$y_{kl} \in \{0,1\}$: 1, if the delivery request from k to l is accepted; 0 otherwise
θ_{ij} : the total flow on arc $(i, j) \in A$, tons
(i.e., the load carried by the vehicle directly from *i* to *j*)
s_i: sequence number for location $i \in$ V

2.2 Original Node-Arc Formulation

Node-arc decision variables
$z_{kl,ij}$ 1, if the delivery from k to l is performed via arc $(i, j) \in A$; 0 otherwise

Node-arc formulation:
Maximize $p\left[\sum_{(k,\,l)\in A} d_{kl} w_{kl} y_{kl}\right] - c\sum_{(i,\,j)\in \mathrm{A}} d_{ij}\theta_{ij} - cv\sum_{(i,\,j)\in A} d_{ij} x_{ij}$ (1)

Subject to:

$\sum_{j\in V} z_{kl,kj} = y_{kl} \quad \forall (k,l) \in A$ (2a)

$\sum_{i\in V} z_{kl,il} = y_{kl} \quad \forall (k,l) \in A$ (2b)

$\sum_{i\in V,\,(i,\,a)\in A} z_{kl,ia} = \sum_{j\in V,\,(a,\,j)\in A} z_{kl,aj} \quad \forall (k,\ l) \in A, a \in V\backslash\{k,l\}$ (2c)

$\sum_{(k,\,l)\in A} z_{kl,ij} \leq M x_{ij} \quad \forall (i,\ j) \in A$ (2d)

$$\sum_{(1,j)\in A} x_{1,j} = 1 \quad (2e)$$

$$\sum_{(i,n)\in A} x_{i,n} = 1 \quad (2f)$$

$$\sum_{i\in V\setminus\{k,\,n\}} x_{ik} = \sum_{j\in V\setminus\{1,\,k\}} x_{kj} \quad \forall k \in V\setminus\{1,n\} \quad (2g)$$

$$\sum_{i\in V\setminus\{k,\,n\}} x_{ik} \leq 1 \quad \forall k \in V\setminus\{1,n\} \quad (2h)$$

$$\sum_{(i,\,j)\in A} d_{ij}x_{ij} \leq D \quad (2i)$$

$$\theta_{ij} = \sum_{(k,l)\in A} w_{kl}z_{kl,ij} \quad \forall (i,j) \in A \quad (2j)$$

$$\theta_{ij} \leq Q \quad \forall (i,j) \in A \quad (2k)$$

$$s_i - s_j + (n+1)x_{ij} \leq n \quad \forall (i,\,j) \in A \quad (2l)$$

$$x_{ij} \in \{0,\,1\} \quad \forall (i,\,j) \in A \quad (2m)$$

$$y_{kl} \in \{0,\,1\} \quad \forall (k,\,l) \in A \quad (2n)$$

$$z_{kl,ij} \in \{0,\,1\} \quad \forall (k,\,l) \in A, (i,\,j) \in A \quad (2o)$$

Formulation Explanation

Objective (1): to maximize net total profit, which is equal to revenue from accepted delivery requests minus travel costs related to delivery requests (cargo-carrying costs) and the vehicle-related travel cost.

Constraints:

Constraints (2a) and to (2b) define the relation between y_{kl} and $z_{kl,ij}$. Constraints (2a) and (2b) are to ensure that if the request from k to l is satisfied, then the vehicle must stop at both location k and location l. Constraint (2c) states that the inbound and outbound traffic flows of a location for the delivery from *k* to *l* should be equal. Constraint (2d) defines the relation between x_{ij} and $z_{kl,ij}$. Constraints (2e) and (2f) make sure that the vehicle will start from location 1 and end at location n. Constraint (2g) states that the inbound and outbound traffic flows of a location should be equal (i.e., if the vehicle stops at location $k \in V \in \{1,n\}$, then it must leave that location) Constraint (2h) states that each location can be visited once at most. Constraint (2i) states that the total length of backhaul trip must not exceed *D*, the maximum allowed distance. Constraint (2j) gives the total load of the vehicle traveling on the arc (i,j). Constraint (2k) enforces the vehicle capacity limit. Constraint (2l) is the so-called MTZ subtour elimination constraint proposed by Miller, Tucker, and Zemlin (1960).

2.3 Original Triples Formulation

Notation for Triples Formulation

T: set of node triples, $\{(i, j, k): i \in V\backslash\{n\}, j \in V\backslash\{1, i\}, k \in V\backslash\{1, n, i, j\}\}$

Triples decision Variables:

$u_{ij}^{k} \geq 0, (i, j, k) \in T$: denotes the tons of cargo transported from node i to node j through node k (i.e., the flow from i to j diverted through k) followed by a path from k to j. Note that the triples formulation given below shares the objective function, and some of the same variables and constraints with the node-arc formulation (those with no labels). The constraints with labels are unique to the triples formulation.

Triples formulation:

Maximize $p\left[\sum_{(k, l)\in A} d_{kl} w_{kl} y_{kl}\right] - c\sum_{(i, j)\in \mathrm{A}} d_{ij}\theta_{ij} - cv\sum_{(i, j)\in A} d_{ij} x_{ij}$

$\sum_{(1,j)\in A} x_{1,j} = 1$

$\sum_{(i,n)\in A} x_{i,n} = 1$

$\sum_{i\in V\backslash\{k, n\}} x_{ik} = \sum_{j\in V\backslash\{1, k\}} x_{kj}, \quad \forall k \in V\backslash\{1, n\}$

$\sum_{(i, j)\in A} d_{ij} x_{ij} \leq D$

$s_i - s_j + (n+1)x_{ij} \leq n \quad \forall (i, j) \in \mathrm{A}$

$\theta_{ij} = w_{ij} y_{ij} + \sum_{(i, k, j)\in T} u_{ik}^{j} + \sum_{(k, j, i)\in T} u_{kj}^{i} - \sum_{(i, j, k)\in T} u_{ij}^{k}, \ \forall (i, j) \in A$ (3a)

$\theta_{ij} \leq Q x_{ij} \ \forall (i, j) \in A$ (3b)

$u_{ij}^{k} \leq Q x_{ik} \ \forall (i, j, k) \in T$ (3c)

$u_{ij}^{k} \geq 0 \ \forall (i, j, k) \in T$ (3d)

$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in A$

$y_{kl} \in \{0, 1\} \quad \forall (k, l) \in A$

Formulation Explanation

The Objective function is the same as that of the node-arc formulation.

Constraints Unique to the Triples Formulation:

Constraint (3a) gives the total load of the vehicle traveling on arc (i, j). Constraint (3b) defines the relationship between the arc flow and x_{ij} variables Constraint (3c) is the linking constraint in order to force (*i*, *k*) to be an arc on the vehicle's route if variable u_{ij}^{k} is positive. Constraint (3d) states that the triples flow must be nonnegative.

2.4 Example Solutions

Recall that in the example solution in Figure 1, the vehicle follows the route corresponding to $x_{13} = x_{35} = x_{56} = 1$, and accepts the delivery requests corresponding to $y_{13} = y_{15} = y_{16} = y_{35} = y_{36} = y_{56} = 1$. The following subsections show how each formulation represents the movement of cargo along the route as multicommodity flow.

2.4.1 Node-Arc Representation of Solution in Figure 1

$z_{13,13} = 1,$
$z_{15,13} = z_{15,35} = 1,$
$z_{16,13} = z_{16,35} = z_{16,56} = 1,$
$z_{35,35} = 1,$
$z_{36,35} = z_{36,56} = 1,$
$z_{56,56} = 1.$

$\theta_{13} = w_{13}z_{13,13} + w_{15}z_{15,13} + w_{16}z_{16,13} = 0.2 + 0.5 + 0.3 = 1$ ton,
$\theta_{35} = w_{15}z_{15,35} + w_{16}z_{16,35} + w_{35}z_{35,35} + w_{36}z_{36,35} = 0.5 + 0.3 + 0.6 + 0.6 = 2$ tons,
$\theta_{56} = w_{16}z_{16,56} + w_{36}z_{36,56} + w_{56}z_{56,56} = 0.3 + 0.6 + 0.8 = 1.7$ tons.

2.4.2 Triples Representation of Solution in Figure 1

$u_{15}^{3} = w_{15} = 0.5$ tons,
$u_{16}^{3} = w_{16} = 0.3$ tons,
$u_{36}^{5} = w_{16} + w_{36} = 0.3 + 0.6 = 0.9$ tons.

Note that the delivery requests from 1 to 3, 3 to 5, and 5 to 6 are sent as "direct flow" and therefore not represented by triples variables. The resulting arc flows are

$\theta_{13} = w_{13} + u_{15}^{3} + u_{16}^{3} = 0.2 + 0.5 + 0.3 = 1$ ton,
$\theta_{35} = w_{35} + u_{15}^{3} + u_{36}^{5} = 0.6 + 0.5 + 0.9 = 2$ tons,
$\theta_{56} = w_{56} + u_{36}^{5} = 0.8 + 0.9 = 1.7$ tons.

3. DESIGN AND ANALYSIS METHODS OF BPMP EXPERIMENTS

In this section we summarize the process of how we design our experiments and develop a multi-criteria Composite Index Method (CIM) to evaluate the results.

3.1 Data Generation

For our study we generate ten problem instances for each value of n = 10, 20, 30, 40, and 50. Following Yu and Dong (2013) and Dong (2015), we assume that the price for delivery service is p = \$1.20 per mile per ton. The travel cost is c = \$1.00 per mile per ton. The maximum time allowed for the backhaul trip is 20 hours and the capacity of the vehicle is Q = 50 tons. The average traveling speed of the vehicle is 50 miles per hour and so the time constraint of 20 hours is equivalent to a distance constraint of D = 1,000 miles. The weight of vehicle itself is v = 5 tons. The remainder of this subsection describes the process for randomly generating the location-to-location distance and weight parameters, d_{ij} and w_{ij}. This process was related to us by Dong (2019); it was not provided in Yu and Dong (2013) and Dong (2015).

The delivery request between two different nodes (in tons) is generated by multiplying Q by a uniform random variable on the range [0, 1] and rounding the result to one decimal place. In this way we can ensure that the demands are randomly and uniformly distributed between zero and the vehicle capacity.

To ensure that all of the randomly generated locations are reachable without violating the distance constraint, we select points inside an ellipse that has nodes 1 and n as its foci. Specifically, we place node 1 and node n at points (500, 250), and (500, 750) in the X-Y plane, respectively as shown in Figure 2. We then select n-2 points at random from inside the ellipse to represent the subset of potential intermediate locations on the vehicle's route from its starting location to its depot.

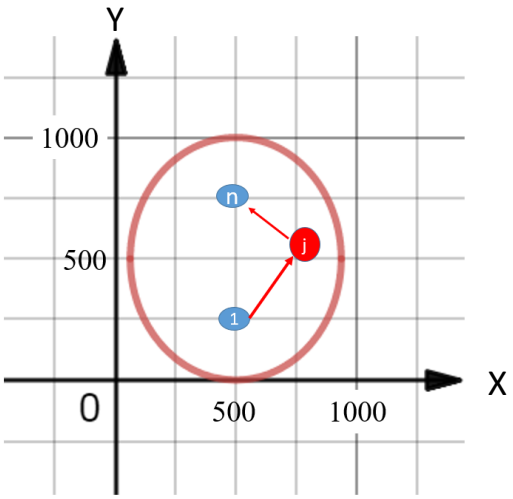

Figure 2. BPMP location-generation diagram.

By construction (i.e., $d_{1j} + d_{jn} \leq 1{,}000$), the vehicle can potentially visit any point j on or inside the ellipse within the time limit and cannot visit any point outside the ellipse. To generate a random point inside the ellipse we start by selecting a y value at random from the range [0, 1,000]. It follows that for the given Y value the points (X_1, Y) and (X_2, Y) are on the ellipse where $X_1 = 500 - 250\sqrt{3 - \frac{3*(Y-500)^2}{500^2}}$ and $X_2 = 500 + 250\sqrt{3 - \frac{3*(Y-500)^2}{500^2}}$ [Open Math Reference 2019]. Thus, we randomly sample the uniform distribution on (X_1, X_2) to generate the X coordinate corresponding to Y. After randomly generating n-2 points inside the ellipse, we let d_{ij} be the Euclidean distance between the corresponding points i and j rounded to three decimal places.

3.2 Computing Environment

The computations reported in Sections 4, 5, and 6 were performed on the SMU Lyle School's general use Linux machines with the specifications listed below. The formulations were implemented in AMPL 10.00 and solved with CPLEX 12.6.0.0. We used the default settings for AMPL and CPLEX except where specified.

Table 1. Computer Hardware Specifications

Make/Model	Dell R730
Processor	Dual 12 Core Intel Xeon@2.6GHz
RAM	320GB

3.3 Performance Evaluation Using Composite Index Method (CIM)

Following the long-standing standard practice in the literature, the plan for this study was to use solution time as the performance measure for comparing the node-arc and triples formulations of BPMP. However, now that computing environments like ours that support multiple users, and take advantage of multiple processors and multiple threads have become commonplace, measuring solution time is no longer straight-forward. Furthermore, as is typically the case, we found that there is often a "crossover point" in problem instance size below which one approach is generally "faster" than another, but above which the second approach is faster. In this situation the second approach would usually be favored because the emphasis in the literature is on solution time as a function of problem instance size. In this study, however, we consider the practical question of making a recommendation to a user who frequently solves problems that range in size around the crossover point, and propose a multi-criteria approach to comparing competing solution approaches. In this section, we develop a Composite Index Method (CIM), which considers several weighted performance measure factors and calculates a single real number (a composite index) to measure the relative performance of two competing solution approaches.

3.3.1 Composite Index for a Given Problem Size

There are three kinds of "solution time" in the CPLEX output: "**CPU time"**, "**real time"**, and "**ticks**". CPU time is a measure of the total time used by CPLEX to find an optimal solution; it is the total time used by all threads. Real time (also called wall clock time) is the time that elapsed during the CPLEX run. Both measures can vary noticeably between runs with identical input on identical hardware. Therefore, we solve each problem instance three times in each experiment and report the average CPU and real time over the three runs. The tick metric, also called deterministic time, is a proprietary measure of computation effort based on counting the number of instructions executed by the CPLEX solver and therefore shows no variation between multiple runs with the same inputs on a given hardware configuration.

For each of the time measures describe above, we report **speedup** to compare the solution time of two models, model 1 versus model 2. Speedup is defined as the ratio

Speedup = Model 1 solution time/Model 2 solution time

If "Speedup">1, model 2 is solved "Speedup" times faster than model 1; if "Speedup"=1, model 2 has the same solution time as model 1, and if "Speedup"<1, model 1 is solved "1/Speedup" times faster than model 2 (i.e., "Speedup" times slower than model 2)..

Due to the fact that CPU and real time are not completely reproducible, we suggest that neither one should be the sole basis for comparing solution approaches. Typically, ticks and real time are positively correlated (as are ticks and CPU time), however there doesn't appear to be a fixed relationship between ticks and the two time measures. For this reason, we cannot use ticks as the single index to compare two models either.

In our experience, customers who use a model to solve a real world problem are much more concerned about real time as a performance measure than CPU time, and are often unaware of the tick measure. For our purposes, however, the reproducibility of the tick metric is quite important. Therefore, we adopt the following weights to each type of time speedup.

Table 2. Weights for Time Speedup

CPU	ω_c	6
Real Time	ω_r	8
Ticks	ω_t	8

For a given problem size (i.e., number of nodes, n) and timing measure (CPU time, real time, or ticks), we calculate a **composite index** based on a weighted combination of the minimum, median, mean, and maximum speedups among 10 instances. Thus, we obtain three composite indices: $CI_n(C)$, $CI_n(R)$, $CI_n(t)$ for CPU time, real time, and ticks, respectively. To calculate these indices we denote the minimum, mean, median, and maximum speedups in CPU time by C_{min}, C_{mean}, C_{median}, and C_{max}, respectively, and define R_{min}, R_{mean}, R_{median}, R_{max}, t_{min}, t_{mean}, t_{median}, and t_{max} as the corresponding speedups for real time and ticks. Additionally, we define ω_{min}, ω_{mean}, ω_{median},

and ω_{max} for different weighting of minimum, mean, median and maximum statistics. Using this notation, the three composite indices are calculated as follows:

$CI_n(C) = (\omega_{min}*C_{min}+\omega_{mea}*C_{mean}+\omega_{med}*C_{median}+\omega_{max}*C_{max})/(\omega_{min}+\omega_{mea}+\omega_{med}+\omega_{max})$,
$CI_n(R) = (\omega_{min}*R_{min}+\omega_{mea}*R_{mean}+\omega_{med}*R_{median}+\omega_{max}*R_{max})/(\omega_{min}+\omega_{mea}+\omega_{med}+\omega_{max})$,
$CI_n(t) = (\omega_{min}*t_{min}+\omega_{mea}*t_{mean}+\omega_{med}*t_{median}+\omega_{max}*t_{max})/(\omega_{min}+\omega_{mea}+\omega_{med}+\omega_{max})$.

Next, we calculate a **composite index**, CI_n, for problem size *n* as a weighted combination of indices $CI_n(C)$, $CI_n(R)$, and $CI_n(t)$:

$$CI_n = (\omega_c * CI_n(C) + \omega_r * CI_n(R) + \omega_t * CI_n(t))/(\omega_c + \omega_r + \omega_t).$$

Table 3 summarizes the process of calculating CI*n* for an experiment with 10 problem instances.

Table 3. Composite Index Calculation Process for 10 Instances of Size *n*

Problem Instance	Speedup		
	CPU Time	Real Time	Ticks
1	C1	R1	t1
2	C2	R2	t2
3	C3	R3	t3
4	C4	R4	t4
5	C5	R5	t5
6	C6	R6	t6
7	C7	R7	t7
8	C8	R8	t8
9	C9	R9	t9
10	C10	R10	t10
Min	C_{min} = min(C1, C2…, C10)	R_{min} = min(R1,R2 …, R10)	t_{min} = min(t1, …, t10)
Mean	C_{mean}= average(C1, C2…, C10)	R_{mean}= average(R1, R2…, R10)	t_{mean}= average(t1, t2…, t10)
Median	C_{median}= median(C1, C2…, C10)	R_{median}= median(R1, R2…, R10)	t_{median}= median(t1, t2…, t10)
Max	C_{max}= max(C1, C2…, C10)	R_{max}= max(R1, R2…, R10)	t_{max}= max(t1, t2…, t10)
Composite Index	$CI_n(C)$= $(\omega_{min}*C_{min}+\omega_{mea}*C_{mean}+\omega_{med}*C_{median}+\omega_{max}*C_{max})/(\omega_{min}+\omega_{mea}+\omega_{med}+\omega_{max})$	$CI_n(R)$= $(\omega_{min}*R_{min}+\omega_{mea}*R_{mean}+\omega_{med}*R_{median}+\omega_{max}*R_{max})/(\omega_{min}+\omega_{mea}+\omega_{med}+\omega_{max})$	$CI_n(t)$= $(\omega_{min}*t_{min}+\omega_{mea}*t_{mean}+\omega_{med}*t_{median}+\omega_{max}*t_{max})/(\omega_{min}+\omega_{mea}+\omega_{med}+\omega_{max})$
	$CI_n=(\omega_c*CI_n(C)+\omega_r*CI_n(R)+\omega_t CI_n(t))/(\omega_c+\omega_r+\omega_t)$		

3.3.2 Grand Composite Index for an Experiment with Multiple Problem Sizes

We solve ten BPMP instances for each of the five problem size n = 10, 20, 30, 40, 50. So we report five composite indices of speedups: CI_{10}, CI_{20}, CI_{30}, CI_{40}, and CI_{50}, which are then transformed into the **Grand Composite Index (GCI)** using problem-size weights ω_n. GCI is the final index we use to compare two models, and is defined as a weighted average of the CI_n values for all sizes of the problem:

$$GCI = (\omega_{10}*CI_{10} + \omega_{20}*CI_{20} + \omega_{30}*CI_{30} + \omega_{40}*CI_{40} + \omega_{50}*CI_{50})/(\omega_{10} + \omega_{20} + \omega_{30} + \omega_{40} + \omega_{50}).$$

In our experiments, we use the weight parameters in Table 4 for the node-arc formulation and those in Table 5 for triples formulation, respectively. The difference in the problem-size weights is due to the fact that solving problem instances with more than 30 nodes using the node-arc model turned out to be impractical, while we could easily solve 50-node instances with the triples model. The selection principle of the weight parameter values is to try to match the results with our intuitive judgement of a relative ranking to apply techniques to a hypothetical logistics company's model as best as possible.

Table 4. Weight Parameters for the Node-Arc Model

Node-Arc Weight Parameters		
Min	ω_{min}	0.5
Median	ω_{med}	40
Max	ω_{max}	0.5
Mean	ω_{mea}	10
CPU	ω_c	6
Ticks	ω_t	8
Real Time	ω_r	8
10-node	ω_{10}	1
20-node	ω_{20}	10
30-node	ω_{30}	12

Table 5. Weight Parameters for the Triples Model

Triples Weight Parameters		
Min	ω_{min}	0.5
Median	ω_{med}	40
Max	ω_{max}	0.5
Mean	ω_{mea}	10
CPU	ω_{c}	6
Ticks	ω_{t}	8
Real Time	ω_{r}	8
10-node	ω_{10}	6
20-node	ω_{20}	10
30-node	ω_{30}	13
40-node	ω_{40}	14
50-node	ω_{50}	16

3.3.3 Using GCI to Compare Models and Evaluate Techniques

In Sections 4 and 5 we use the GCI to evaluate the efficacy of various techniques (cuts, branching rules, etc.) designed to improve CPLEX's performance using the node-arc and triples models given in Section 2. Using "model 1" to refer to a baseline solution approach and "model 2" to refer to the application of a particular technique to model 1. We recommend adopting the technique if GCI > 1 and say, for convenience, that the model 2 is "GCI times faster" than model 1. We recommend not adopting the technique if GCI ≤ 1.

3.3.4 Using LP upper bound improvement to compare the strength of models

We denote the upper bounds on profit obtained from the LP relaxations of model 1 and model 2 mentioned above as LP1 and LP2, and define **LP improvement** as (LP1-LP2)/LP1. Since the BPMP is a maximization problem, the smaller the LP upper bound, the stronger the model. Thus, a positive LP improvement indicates that the technique applied in model 2 makes model 1 stronger.

4. ENHANCING THE NODE-ARC FORMULATION

In this section we present experimental results with nine techniques for enhancing the node-arc formulation. These techniques were selected and informally ranked by effectiveness from a larger set of candidates after preliminary experiments that we performed prior to developing the CIM. Before applying the first of the nine techniques, we establish an "incumbent" enhanced node-arc formulation by determining a tight Big-*M* value for the *x*-*z* linking constraint set (2d). We then apply the techniques sequentially according to the ranking from our preliminary experiments. If a particular technique in the sequence is found to improve performance based on the CIM, the combination of the incumbent with the technique becomes the new incumbent. Otherwise, the technique is not adopted and the incumbent remains as it is. Note that CPU and real time are reported in seconds, and LP bounds are scaled by $2,500 throughout this report.

4.1 Tightening the Big-M Value

Before running the node-arc model experiments, first we need to decide the value of *M* in (2d) of Section 2.2, $\sum_{(k,\,l)\in A} z_{kl,ij} \le M x_{ij} \;\; (i,\,j) \in A$. Yu and Dong (2013) did not discuss the value of *M*. Here we prove that a value of $M = \frac{n^2-n}{2}$, where *n* is the total number of nodes in the network, is sufficient.

Proof:

Observe that the left-hand side of (2d) is less than or equal to the total number of accepted delivery requests, $\sum_{(k,l)\in A} y_{kl}$.

$$\sum_{(k,\,l)\in A} z_{kl,ij} \le M x_{ij} \;\; (i,\,j) \in A$$

Due to constraints (2h) and (2l), the vehicle can stop at most once at each node. Suppose that the vehicle visits every node and, without loss of generality, follows the route 1, 2, 3, …, *n*. At most, the vehicle can accept *n*-1 requests from node 1, *n*-2 from node 2, etc. Thus, the maximum number of requests that the vehicle can accept is

$$\sum_{k=1}^{n}(n-k) = \sum_{k=1}^{n-1} k = \frac{(n-1)(n)}{2} = \frac{n^2-n}{2}.$$

4.2 Initial Incumbent Formulation

In this section we present the results for our initial incumbent, the original node-arc model with the tightened Big-*M* value. The model was solved three times for each problem instance. The results are shown in Table 6.

Table 6.a Test Results of Original Node-Arc Model for n =10.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	5.22	3.54	2.88	3.88	1.09	0.89	1.03	1.00	58.64	345.86
02	4.39	4.75	4.08	4.40	1.16	0.93	1.28	1.12	61.42	225.56
03	3.83	4.29	4.05	4.06	0.94	0.76	0.85	0.85	57.95	291.53
04	2.95	2.86	4.47	3.42	1.00	0.80	1.02	0.94	68.63	252.59
05	3.54	2.23	2.84	2.87	0.82	0.66	0.81	0.76	77.40	237.66
06	6.42	6.18	6.75	6.45	2.62	2.18	2.87	2.56	63.48	645.50
07	6.99	4.97	3.79	5.25	1.42	1.22	1.43	1.36	63.24	496.25
08	4.20	4.68	4.88	4.59	1.53	1.27	1.55	1.45	62.24	499.80
09	6.61	3.88	3.12	4.54	1.06	0.83	1.03	0.97	66.86	214.21
10	3.02	1.64	2.17	2.28	0.52	0.37	0.61	0.50	67.96	120.06
Min	2.95	1.64	2.17	2.28	0.52	0.37	0.61	0.50	57.95	120.06
Mean	4.72	3.90	3.90	4.17	1.22	0.99	1.25	1.15	64.78	332.90
Median	4.30	4.08	3.92	4.23	1.08	0.86	1.03	0.99	63.36	272.06
Max	6.99	6.18	6.75	6.45	2.62	2.18	2.87	2.56	77.40	645.50

Table 6.b Test Results of Original Node-Arc Model for n = 20.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	12,654	12,310	12,172	12,379	1,937	1,896	1,734	1,856	316	1,244,880
02	1,745	1,757	1,716	1,739	355	372	344	357	330	272,703
03	2,457	2,544	2,393	2,465	497	536	467	500	291	382,350
04	24,954	26,053	27,313	26,106	1,891	2,044	1,943	1,960	293	1,211,197
05	2,964	3,036	2,981	2,994	632	681	628	647	323	583,158
06	32,635	35,586	34,566	34,262	1,720	1,859	1,778	1,786	277	1,005,079
07	2,760	2,770	2,615	2,715	502	512	482	499	300	422,516
08	5,873	5,888	5,738	5,833	845	857	806	836	272	674,853
09	28,620	32,115	28,746	29,827	2,007	2,226	1,967	2,067	299	1,318,085
10	17,096	17,745	18,202	17,681	1,571	1,749	1,606	1,642	340	1,091,225
Min	1,745	1,757	1,716	1,739	355	372	344	357	272	272,703
Mean	13,176	13,980	13,644	13,600	1,196	1,273	1,175	1,215	304	820,605
Median	9,263	9,099	8,955	9,106	1,208	1,303	1,206	1,239	300	839,966
Max	32,635	35,586	34,566	34,262	2,007	2,226	1,967	2,067	340	1,318,085

As shown in Tables 6.a and 6.b, we were only able to solve 10-node and 20-node instances with the original node-arc model. Therefore, we list the results only for 10 and 20-node instances, and there are no speedups yet (no techniques applied yet). We can see that the median average real time for the 10-node instances was about 1 second, and the median average real time for the 20-node instances was about 20 minutes.

4.3 Technique 1: Conditional Arc-Flow

The original node-arc model (Yu and Dong 2013) uses constraint (2k), $\theta_{ij} \leq Q$, to ensure that the total amount of flow, θ_{ij}, on arc (i, j) is less than or equal to the vehicle capacity, Q. Notice that if the vehicle does not travel on arc (i, j), there should be no flow on the arc (i.e., if $x_{ij} = 0$, then $\theta_{ij} = 0$). If the vehicle does travel on arc (i, j), the maximum flow on the arc is Q, (i.e., if $x_{ij} = 1$, then $\theta_{ij} \leq Q$). Therefore, the arc flow constraints (2k) can be replaced by the following constraint set which we call *conditional arc-flow*

$$\theta_{ij} \leq Q x_{ij} \qquad \forall (i, j) \in A \qquad (2p).$$

Yu and Dong (2013) were unable to solve 30-node instances with the original node-arc model. We had a similar experience in our preliminary tests. After applying conditional arc-flow we can solve 30-node instances easily, but it is difficult to solve 40-node instances. Therefore, we tested this technique only on 10-, 20-, and 30-node instances. Table 7 gives detailed test results of three runs after applying the new technique on 10-, 20-, and 30-node instances. The complete speedup summary is given in Table 8. Table 9 gives the CI and GCI. Speedups in bold are greater than 1.

Table 7.a Test Results of Incremental Effect of Conditional Arc-Flow for n =10.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	1.42	1.33	1.46	1.40	0.61	0.57	0.63	0.60	2.00	235.67
02	3.47	3.79	1.90	3.05	0.62	0.59	0.58	0.60	2.00	248.86
03	2.43	3.13	5.15	3.57	0.71	0.79	0.82	0.77	2.94	278.19
04	4.06	2.14	2.13	2.78	0.63	0.61	0.62	0.62	5.01	188.38
05	0.87	0.93	0.92	0.90	0.43	0.45	0.43	0.44	6.50	184.29
06	3.23	4.94	4.93	4.37	1.00	0.94	0.93	0.96	8.29	247.70
07	2.90	2.36	4.09	3.12	0.92	0.86	0.90	0.89	3.47	384.78
08	3.39	4.64	6.44	4.82	0.92	0.90	1.02	0.95	3.87	366.32
09	2.70	2.61	2.64	2.65	0.55	0.52	0.46	0.51	6.29	164.87
10	1.51	1.59	2.37	1.83	0.22	0.26	0.31	0.26	9.72	74.61
Min	0.87	0.93	0.92	0.90	0.22	0.26	0.31	0.26	2.00	74.61
Mean	2.60	2.75	3.20	2.85	0.66	0.65	0.67	0.66	5.01	237.37
Median	2.80	2.49	2.50	2.91	0.63	0.60	0.63	0.61	4.44	241.69
Max	4.06	4.94	6.44	4.82	1.00	0.94	1.02	0.96	9.72	384.78

Table 7.b Test Results of Incremental Effect of Conditional Arc-Flow for $n = 20$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	1,306	1,258	1,266	1,276	96	100	97	98	16	54,992
02	720	701	702	708	63	63	64	63	17	37,316
03	1,974	1,689	1,729	1,797	143	146	134	141	22	81,335
04	4,945	3,848	3,836	4,210	358	362	318	346	18	202,725
05	450	409	408	422	94	94	92	93	22	70,847
06	2,152	1,752	1,764	1,890	196	200	182	193	15	119,363
07	1,243	1,105	1,120	1,156	81	81	77	80	18	39,778
08	1,189	1,080	1,010	1,093	74	76	68	73	13	35,113
09	673	630	620	641	127	130	125	128	19	96,582
10	811	759	740	770	123	125	119	122	24	85,867
Min	450	409	408	422	63	63	64	63	13	35,113
Mean	1,546	1,323	1,320	1,396	136	138	128	134	19	82,392
Median	1,216	1,093	1,065	1,125	110	112	108	110	18	76,091
Max	4,945	3,848	3,836	4,210	358	362	318	346	24	202,725

Table 7.c Test Results of Incremental Effect of Conditional Arc-Flow for $n = 30$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	236,367	237,258	212,689	228,771	12,301	12,474	11,552	12,109	33	3,442,157
02	97,710	92,781	81,972	90,821	5,958	5,780	5,382	5,707	32	2,203,579
03	155,158	156,250	137,272	149,560	8,349	8,451	7,660	8,153	32	2,738,406
04	174,107	171,060	152,215	165,794	9,580	9,439	8,757	9,259	45	3,274,659
05	78,981	77,249	68,914	75,048	5,301	5,241	4,941	5,161	24	2,074,290
06	195,855	195,754	176,959	189,523	10,264	10,146	9,557	9,989	41	3,389,669
07	57,534	56,502	50,478	54,838	4,506	4,485	4,271	4,421	27	2,104,806
08	324,709	332,816	298,199	318,575	17,912	18,534	17,041	17,829	22	5,072,975
09	99,423	106,375	87,302	97,700	5,697	6,221	5,228	5,716	22	2,177,300
10	153,816	154,163	131,121	146,367	7,287	7,562	6,499	7,116	32	2,383,682
Min	57,534	56,502	50,478	54,838	4,506	4,485	4,271	4,421	22	2,074,290
Mean	157,366	158,021	139,712	151,700	8,716	8,833	8,089	8,546	31	2,886,152
Median	154,487	155,207	134,197	147,963	7,818	8,006	7,079	7,635	32	2,561,044
Max	324,709	332,816	298,199	318,575	17,912	18,534	17,041	17,829	45	5,072,975

Table 8. Summary of Incremental Effect of Conditional Arc Flow Constraints

	Speedup			LP Upper Bound Improvement
	Ave. CPU Time	Ticks	Ave. Real Time	
n = 10				
Min	0.95	0.91	**1.10**	85.69%
Mean	**1.68**	**1.42**	**1.74**	92.41%
Median	**1.46**	**1.32**	**1.71**	93.24%
Max	**3.17**	**2.61**	**2.67**	96.74%
n = 20				
Min	**1.37**	**4.70**	**3.55**	93.32%
Mean	**12.21**	**11.35**	**9.74**	93.94%
Median	**6.65**	**9.52**	**8.10**	93.95%
Max	**46.52**	**22.64**	**19.01**	95.08%

Table 9. CI and GCI of Conditional Arc-Flow Constraints

n	CPU	Ticks	Real Time	CI	GCI
10	**1.52**	**1.35**	**1.72**	**1.53**	**8.24**
20	**8.08**	**9.96**	**8.48**	**8.91**	
30	n/a	n/a	n/a	n/a	

Conclusion: After applying technique 1, conditional arc-flow, the grand composite index of speedups (GCI) was 8.24, which means, on average, the model with conditional arc-flow was solved 8.24 times faster than the original model. Therefore, we adopted technique 1, replacing constraint set (2k) with the conditional arc-flow constraints (2p).

4.4 Technique 2： Relax Node-Degree Constraints

Yu and Dong (2013) used the following **node-degree** cuts to ensure that the vehicle visits each location at most once:

$$\sum_{i \in V \backslash \{k,\, n\}} x_{ik} \leq 1 \quad k \in V \backslash \{1, n\} \qquad \text{(2h)}$$

At the same time the MTZ subtour elimination constraints (2l) also ensure that vehicle visits each node at most once in an integer solution. Therefore, we can relax the node-degree constraints without losing validity of the integer model (the node-degree cuts can be violated in solutions to the LP relaxations). Table 10 gives detailed test results of three runs after applying the new technique (dropping/relaxing (2h)) on 10-, 20-, and 30-node instances. Table 11 shows the effect of the relaxing the node-degree cuts on all of the 10-, 20-, and 30-node problem instances. Table 12 gives the CI and GCI. Speedups greater than 1 are in bold.

Table 10.a Test Results of Incremental Effect of Relax Node-Degree for $n = 10$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	1.10	1.17	1.16	1.15	0.49	0.53	0.51	0.51	2.00	197.98
02	2.20	2.89	6.01	3.70	0.44	0.42	0.56	0.47	2.00	153.40
03	7.34	4.71	4.72	5.59	1.51	0.85	1.24	1.20	2.94	251.73
04	4.87	2.79	2.04	3.23	0.69	0.78	0.76	0.74	5.01	195.15
05	7.40	3.65	6.28	5.78	0.60	0.47	0.52	0.53	6.50	133.75
06	8.66	3.00	7.26	6.31	1.26	0.98	1.19	1.14	8.60	334.68
07	4.86	2.52	7.60	4.99	0.97	0.79	1.04	0.93	3.47	187.98
08	8.81	2.05	7.84	6.23	1.09	0.75	1.05	0.96	3.92	313.69
09	2.35	1.89	3.59	2.61	0.51	0.43	0.53	0.49	6.29	158.90
10	6.53	1.92	1.51	3.32	0.40	0.51	0.35	0.42	9.84	66.23
Min	1.10	1.17	1.16	1.15	0.40	0.42	0.35	0.42	2.00	66.23
Mean	5.41	2.66	4.80	4.29	0.80	0.65	0.78	0.74	5.06	199.35
Median	5.70	2.65	5.37	4.35	0.65	0.64	0.66	0.64	4.46	191.57
Max	8.81	4.71	7.84	6.31	1.51	0.98	1.24	1.20	9.84	334.68

Table 10.b Test Results of Incremental Effect of Relax Node-Degree for $n = 20$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	1,354	1,373	1,368	1,365	82	83	82	82	18	42,788
02	743	732	732	736	78	77	77	77	19	49,490
03	1,203	1,123	1,145	1,157	76	73	75	75	22	38,808
04	7,230	6,503	6,763	6,832	404	376	379	386	20	197,501
05	379	334	350	355	42	40	41	41	24	28,931
06	2,589	2,271	2,339	2,400	189	179	179	182	15	118,673
07	1,055	1,118	1,009	1,061	67	71	64	67	18	32,021
08	901	920	826	882	59	62	56	59	14	29,653
09	1,591	1,498	1,462	1,517	186	198	184	189	19	134,464
10	674	668	644	662	69	72	68	70	24	45,904
Min	379	334	350	355	42	40	41	41	14	28,931
Mean	1,772	1,654	1,664	1,697	125	123	120	123	19	71,823
Median	1,129	1,121	1,077	1,109	77	75	76	76	19	44,346
Max	7,230	6,503	6,763	6,832	404	376	379	386	24	197,501

Table 10.c Test Results of Incremental Effect of Relax Node-Degree for $n = 30$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	150,867	160,878	157,288	156,344	8,626	9,369	8,760	8,919	35	2,927,573
02	315,832	340,140	341,062	332,345	12,468	13,185	13,096	12,917	34	3,904,553
03	51,365	52,736	53,121	52,407	2,959	3,190	3,043	3,064	32	1,504,484
04	114,123	116,978	118,709	116,603	5,349	5,658	5,525	5,511	45	2,141,250
05	107,272	109,664	110,180	109,039	5,080	5,285	5,173	5,179	24	2,082,700
06	176,095	180,395	179,597	178,696	9,602	10,218	9,734	9,851	41	3,083,977
07	118,167	120,838	123,134	120,713	5,163	5,416	5,316	5,298	27	1,889,653
08	103,211	109,225	106,275	106,237	5,308	5,885	5,467	5,553	23	1,874,691
09	42,476	43,265	43,337	43,026	2,301	2,405	2,317	2,341	23	1,142,160
10	102,503	103,896	105,209	103,869	4,576	4,791	4,697	4,688	32	1,836,292
Min	42,476	43,265	43,337	43,026	2,301	2,405	2,317	2,341	23	1,142,160
Mean	128,191	133,802	133,791	131,928	6,143	6,540	6,313	6,332	32	2,238,733
Median	110,698	113,321	114,445	112,821	5,235	5,537	5,391	5,404	32	1,986,176
Max	315,832	340,140	341,062	332,345	12,468	13,185	13,096	12,917	45	3,904,553

Table 11. Summary of Incremental Effect of Relax Node-Degree Constraints

	Speedup			LP Upper Bound Improvement
	Ave. CPU Time	Ticks	Ave. Real Time	
$n = 10$				
Min	0.16	0.74	0.63	-3.74%
Mean	0.74	**1.24**	0.92	-0.63%
Median	0.73	**1.15**	0.90	0.00%
Max	**1.23**	**2.05**	**1.26**	0.00%
$n = 20$				
Min	0.42	0.72	0.67	-11.18%
Mean	1.00	**1.36**	**1.30**	-4.34%
Median	**1.03**	**1.21**	**1.19**	-3.08%
Max	**1.55**	**2.45**	**2.25**	0.00%
$n = 30$				
Min	0.27	0.56	0.44	-5.45%
Mean	**1.49**	**1.42**	**1.62**	-2.05%
Median	**1.42**	**1.24**	**1.44**	-0.55%
Max	**3.00**	**2.71**	**3.21**	0.00%

Table 12. CI and GCI of Relax Node-Degree Constraints

n	CPU	Ticks	Real Time	CI	GCI
10	0.73	**1.17**	0.90	0.95	**1.28**
20	**1.02**	**1.25**	**1.22**	**1.17**	
30	**1.43**	**1.28**	**1.48**	**1.40**	

Conclusion: After applying technique 2, relax node-degree constraints, the grand composite index of speedups (GCI) was 1.28, which means, on average, the model relaxing the constraints was solved 1.28 times faster than the incumbent model. Therefore, we adopted technique 2 and dropped constraint set (2h) from the incumbent.

4.5 Technique 3：Single-Node Demand Cuts

The **single-node demand cuts** state that the total weight of the delivery requests accepted from node *i*, or into node *j*, is at most the vehicle capacity, *Q*.

$$\sum_{j \in V \setminus \{1,i\}} w_{ij} y_{ij} \leq Q \qquad \forall i \in V \setminus \{n\}$$

$$\sum_{i \in V \setminus \{j,n\}} w_{ij} y_{ij} \leq Q \qquad \forall j \in V \setminus \{1\}$$

The above are valid inequalities that are satisfied by any feasible solution because the vehicle cannot simultaneously hold cargoes with total weights more than its capacity *Q*. This condition is not necessarily enforced by solutions to the LP relaxation because of the fractional *y* values. Tables 13, 14, and 15 give the results from applying single-node demand cuts to the incumbent node-arc model.

Table 13.a Test Results of Incremental Effect of Single-Node Demand Cuts for $n = 10$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	3.28	1.78	7.33	4.13	0.62	0.6	0.76	0.66	2.00	205.75
02	5.71	3.71	2.79	4.07	0.54	0.49	0.42	0.48	2.00	162.91
03	2.33	2.60	3.40	2.78	0.84	0.93	0.88	0.88	2.94	279.41
04	6.21	2.67	4.60	4.49	0.73	0.59	0.68	0.67	5.01	218.26
05	0.47	0.54	0.47	0.49	0.26	0.29	0.26	0.27	6.50	129.68
06	8.78	4.57	5.05	6.13	1.13	0.87	0.97	0.99	8.60	287.70
07	9.31	5.98	4.95	6.74	0.85	0.69	0.68	0.74	3.47	184.64
08	1.91	1.75	1.60	1.75	0.78	0.71	0.68	0.72	3.92	287.99
09	9.28	3.83	1.49	4.87	0.77	0.55	0.42	0.58	6.29	164.44
10	5.55	3.96	0.68	3.40	0.37	0.32	0.19	0.29	9.84	95.58
Min	0.47	0.54	0.47	0.49	0.26	0.29	0.19	0.27	2.00	95.58
Mean	5.28	3.14	3.24	3.89	0.69	0.60	0.59	0.63	5.06	201.64
Median	5.63	3.19	3.09	4.10	0.75	0.60	0.68	0.66	4.46	195.20
Max	9.31	5.98	7.33	6.74	1.13	0.93	0.97	0.99	9.84	287.99

Table 13.b Test Results of Incremental Effect of Single-Node Demand Cuts for $n = 20$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	1,501	1,579	1,532	1,537	86	89	88	88	18	39,814
02	775	765	756	765	59	57	57	58	19	29,614
03	2,074	2,136	2,167	2,126	149	153	152	151	22	79,696
04	7,891	8,132	8,443	8,155	501	508	504	504	20	243,567
05	385	394	414	397	55	54	56	55	24	38,703
06	2,148	2,292	2,436	2,292	136	142	148	142	15	77,267
07	1,249	1,283	1,290	1,274	89	91	89	89	18	44,079
08	1,070	1,053	1,077	1,067	72	70	70	70	14	34,261
09	2,042	2,127	2,204	2,124	210	208	209	209	19	135,340
10	625	630	638	631	75	74	75	75	24	49,812
Min	385	394	414	397	55	54	56	55	14	29,614
Mean	1,976	2,039	2,096	2,037	143	145	145	144	19	77,215
Median	1,375	1,431	1,411	1,405	87	90	89	88	19	46,945
Max	7,891	8,132	8,443	8,155	501	508	504	504	24	243,567

Table 13.c Test Results of Incremental Effect of Single-Node Demand Cuts for $n = 30$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	131,326	133,475	129,676	131,492	11,016	10,934	10,484	10,811	35	4,211,599
02	115,556	123,189	120,357	119,701	5,608	5,863	5,790	5,753	34	2,244,072
03	53,661	56,906	56,559	55,708	3,308	3,356	3,296	3,320	32	1,625,743
04	128,082	136,079	133,987	132,716	5,853	6,123	6,034	6,004	45	2,077,826
05	93,717	99,684	98,291	97,231	4,617	4,835	4,760	4,737	24	1,806,019
06	392,279	396,132	393,380	393,930	20,620	20,981	20,295	20,632	41	5,753,277
07	124,600	131,991	129,916	130,954	5,563	5,832	5,668	5,750	27	1,967,062
08	235,304	240,596	232,432	236,514	11,921	12,057	11,407	11,732	23	3,309,807
09	41,558	43,813	43,397	42,923	2,703	2,750	2,727	2,727	23	1,386,778
10	87,632	93,870	90,754	90,752	4,238	4,428	4,266	4,311	32	1,649,855
Min	41,558	43,813	43,397	42,923	2,703	2,750	2,727	2,727	23	1,386,778
Mean	140,371	145,573	142,875	143,192	7,545	7,716	7,473	7,578	32	2,603,204
Median	120,078	127,590	125,017	125,327	5,585	5,847	5,729	5,752	32	2,022,444
Max	392,279	396,132	393,380	393,930	20,620	20,981	20,295	20,632	45	5,753,277

Table 14. Summary of Incremental Effect of Single-Node Demand Cuts

	Speedup			LP Upper Bound Improvement
	Ave. CPU Time	Ticks	Ave. Real Time	
n = 10				
Min	0.28	0.69	0.77	0.00%
Mean	**2.25**	0.97	**1.22**	0.00%
Median	0.94	0.96	**1.21**	0.00%
Max	**11.76**	**1.16**	**1.96**	0.00%
n = 20				
Min	0.54	0.49	0.49	0.00%
Mean	0.86	0.98	0.90	0.18%
Median	0.86	0.89	0.87	0.00%
Max	**1.05**	**1.67**	**1.34**	1.83%
n = 30				
Min	0.45	0.54	0.47	0.00%
Mean	**1.09**	0.95	0.98	0.00%
Median	0.97	0.94	0.92	0.00%
Max	**2.78**	**1.74**	**2.25**	0.00%

Table 15. CI and GCI of Single-Node Demand Cuts

n	CPU	Ticks	Real Time	CI	GCI
10	**1.30**	0.96	**1.21**	**1.15**	0.94
20	0.86	0.91	0.88	0.89	
30	**1.01**	0.95	0.94	0.96	

Conclusion: After applying technique 3, single-node demand cuts, the grand composite index of speedups (GCI) was 0.94, which means that solving the incumbent model was faster. Therefore, we did not adopt technique 3.

4.6 Technique 4：Relax *x*-*z* Linking Constraints

Since $\theta_{ij} = \sum_{(k,l)\in A} w_{kl} z_{kl,ij}$, adopting the conditional arc-flow cuts, $\theta_{ij} \leq Q x_{ij}$, makes the constrains (d2) linking the *x* and *z* variables, $\sum_{(k,l)\in A} z_{kl,ij} \leq M x_{ij}$, redundant. Tables 16, 17, and 18 give the results from relaxing constraint set (2d) in the incumbent node-arc model.

Table 16.a Test Results of Incremental Effect of Relax *x*-*z* Linking for $n = 10$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	2.43	1.28	2.28	2.00	0.67	0.64	0.62	0.64	2.00	250.43
02	3.30	3.45	7.08	4.61	0.82	0.78	0.72	0.77	2.00	96.32
03	3.07	1.73	4.79	3.19	0.73	0.69	0.61	0.68	2.94	190.38
04	4.19	2.48	5.53	4.07	0.76	0.59	0.95	0.77	5.01	199.94
05	0.92	0.90	0.93	0.92	0.39	0.41	0.38	0.39	6.50	109.72
06	4.00	2.03	1.87	2.64	0.98	0.95	0.80	0.91	8.60	326.40
07	4.27	2.44	8.25	4.99	0.88	0.83	0.96	0.89	3.47	272.16
08	3.48	2.30	3.20	2.99	0.71	0.63	0.66	0.67	3.92	229.48
09	4.49	2.74	7.79	5.01	0.65	0.64	0.85	0.71	6.29	163.54
10	2.13	1.41	2.05	1.86	0.25	0.23	0.25	0.24	9.84	41.91
Min	0.92	0.90	0.93	0.92	0.25	0.23	0.25	0.24	2.00	41.91
Mean	3.23	2.08	4.38	3.23	0.68	0.64	0.68	0.67	5.06	188.03
Median	3.39	2.17	3.99	3.09	0.72	0.64	0.69	0.70	4.46	195.16
Max	4.49	3.45	8.25	5.01	0.98	0.95	0.96	0.91	9.84	326.40

Table 16.b Test Results of Incremental Effect of Relax *x*-*z* Linking for $n = 20$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	1,036	1,070	1,059	1,055	71	71	70	70	18	37,045
02	769	802	779	783	52	54	53	53	19	27,438
03	1,687	1,735	1,767	1,730	130	133	132	131	22	74,716
04	6,526	6,560	6,983	6,690	368	374	382	375	20	178,757
05	202	208	213	208	40	40	40	40	24	29,129
06	1,858	1,903	1,980	1,914	137	140	141	139	15	88,623
07	1,064	1,121	1,078	1,088	77	80	78	79	18	41,022
08	780	823	795	799	52	54	52	53	14	26,163
09	1,136	1,126	1,218	1,160	119	121	122	121	19	80,477
10	1,470	1,493	1,545	1,503	94	92	94	93	24	51,420
Min	202	208	213	208	40	40	40	40	14	26,163
Mean	1,653	1,684	1,742	1,693	114	116	116	115	19	63,479
Median	1,100	1,123	1,148	1,124	85	86	86	86	19	46,221
Max	6,526	6,560	6,983	6,690	368	374	382	375	24	178,757

Table 16.c Test Results of Incremental Effect of Relax *x-z* Linking for $n = 30$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	120,138	117,252	118,408	118,599	7,915	7,821	7,772	7,836	35	2,902,468
02	205,899	196,957	199,742	198,350	8,376	8,147	8,304	8,225	34	2,561,425
03	75,956	72,568	74,091	74,205	4,131	3,969	4,128	4,076	32	1,590,695
04	152,491	145,197	146,619	148,102	6,978	6,755	6,812	6,848	45	2,235,008
05	41,576	39,272	40,190	40,346	2,184	2,151	2,140	2,159	24	895,895
06	114,124	111,409	112,341	112,625	6,437	6,593	6,392	6,474	41	2,041,673
07	41,915	41,252	41,678	41,615	2,601	2,561	2,578	2,580	27	1,209,631
08	107,115	103,201	104,597	104,971	5,887	5,775	5,773	5,812	23	1,935,653
09	52,570	48,790	50,296	50,552	2,838	2,780	2,769	2,796	23	1,151,985
10	75,831	72,063	74,851	74,248	3,893	3,806	3,884	3,861	32	1,521,308
Min	41,576	39,272	40,190	40,346	2,184	2,151	2,140	2,159	23	895,895
Mean	98,761	94,796	96,281	96,361	5,124	5,036	5,055	5,067	32	1,804,574
Median	91,536	87,884	89,724	89,610	5,009	4,872	4,950	4,944	32	1,763,174
Max	205,899	196,957	199,742	198,350	8,376	8,147	8,304	8,225	45	2,902,468

Table 17. Summary of Incremental Effect of Relax *x-z* Linking Constraints

	Speedup			LP Upper Bound Improvement
	Ave. CPU Time	Ticks	Ave. Real Time	
$n = 10$				
Min	0.52	0.69	0.61	0.00%
Mean	**1.80**	**1.15**	**1.17**	0.00%
Median	**1.38**	**1.12**	**1.15**	0.00%
Max	**6.29**	**1.59**	**1.77**	0.00%
$n = 20$				
Min	0.44	0.52	0.57	0.00%
Mean	**1.07**	**1.14**	**1.09**	0.00%
Median	**1.06**	**1.12**	**1.08**	0.00%
Max	**1.71**	**1.80**	**1.56**	0.00%
$n = 30$				
Min	0.71	0.95	0.75	0.00%
Mean	**1.49**	**1.30**	**1.32**	0.00%
Median	**1.36**	**1.11**	**1.18**	0.00%
Max	**2.90**	**2.32**	**2.40**	0.00%

Table 18. CI and GCI of Relax *x-z* Linking Constraints

n	CPU	Ticks	Real Time	CI	GCI
10	**1.50**	**1.13**	**1.16**	**1.24**	**1.18**
20	**1.06**	**1.12**	**1.08**	**1.09**	
30	**1.39**	**1.16**	**1.21**	**1.24**	

Conclusion: After applying technique 4, Relax *x-z* Linking, the grand composite index of speedups (GCI) was 1.18, which means, on average, solving the model relaxing *x-z* linking constraints was 1.18 times faster than the incumbent model. Therefore, we adopted technique 4 and dropped constraint set (2d) from the incumbent.

4.7 Technique 5：Branching Priority

In the node-arc model, there are three types of binary variables: x_{ij}, y_{kl}, and $z_{kl,ij}$ which indicate whether the vehicle travels on arc (*i*, *j*), whether the delivery request from node *k* to node *l* is accepted, and whether the accepted demand from node *k* to node *l* is realized via arc (*i*, *j*), respectively. We suspected that prioritizing determining the vehicle's route over deciding which delivery requests to accept would lead to faster solution times. Therefore, we tested solving the problem with a branching rule that stating that *x* variables are branched on before any other binary variables. The results are given in Tables 19, 20, and 21.

Table 19.a Test Results of Incremental Effect of Branching Priority for n =10.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	1.45	3.80	4.95	3.40	0.64	0.77	0.73	0.71	2.00	250.44
02	3.28	5.19	9.40	5.95	0.73	0.62	0.67	0.67	2.00	96.33
03	2.19	3.74	8.23	4.72	0.61	0.56	0.81	0.66	2.94	190.38
04	3.85	3.41	10.57	5.94	0.58	0.64	0.92	0.71	5.01	199.95
05	0.86	0.83	0.64	0.78	0.36	0.34	0.28	0.33	6.50	109.73
06	4.42	5.55	4.78	4.92	0.99	0.87	0.87	0.91	8.60	326.41
07	4.22	4.88	8.21	5.77	0.97	1.00	0.92	0.96	3.47	272.17
08	3.92	5.36	6.52	5.27	0.71	0.79	0.82	0.77	3.92	229.49
09	1.83	6.42	7.47	5.24	0.50	0.68	0.66	0.61	6.29	163.54
10	1.07	2.26	1.37	1.57	0.18	0.27	0.19	0.21	9.84	41.91
Min	0.86	0.83	0.64	0.78	0.18	0.27	0.19	0.21	2.00	41.91
Mean	2.71	4.14	6.21	4.35	0.63	0.65	0.69	0.66	5.06	188.04
Median	2.73	4.34	6.99	5.08	0.63	0.66	0.77	0.69	4.47	195.17
Max	4.42	6.42	10.57	5.95	0.99	1.00	0.92	0.96	9.84	326.41

Table 19.b Test Results of Incremental Effect of Branching Priority for n =20.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	745	816	873	812	54	57	60	57	18	29,107
02	720	811	867	800	43	46	48	46	19	20,459
03	869	981	1,040	963	58	62	65	62	22	30,882
04	1,125	1,297	1,366	1,262	95	104	109	102	20	57,075
05	120	131	127	126	29	31	31	31	24	20,871
06	737	887	799	807	98	110	107	105	15	72,647
07	878	981	1,046	968	58	63	67	62	18	29,078
08	656	687	780	708	40	42	45	42	14	18,475
09	248	293	264	268	45	49	47	47	19	32,132
10	497	559	544	533	51	55	54	54	24	33,485
Min	497	131	127	126	29	31	31	31	14	18,475
Mean	497	744	771	725	57	62	63	61	19	34,421
Median	497	814	833	804	53	56	57	55	19	29,995
Max	497	1,297	1,366	1,262	98	110	109	105	24	72,647

Table 19.c Test Results of Incremental Effect of Branching Priority for n = 30.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	61,218	69,127	70,375	66,907	3,875	4,393	4,522	4,263	35	1,405,157
02	128,243	149,091	143,728	140,354	5,027	5,778	5,687	5,497	34	1,546,461
03	38,096	44,385	43,470	41,983	2,505	2,881	2,935	2,774	32	1,213,322
04	18,483	21,745	21,133	20,454	1,464	1,656	1,662	1,594	45	812,789
05	19,407	22,517	21,996	21,307	1,503	1,670	1,728	1,634	24	838,275
06	33,504	37,132	40,321	36,986	1,917	2,118	2,284	2,106	41	863,673
07	29,945	34,850	34,001	32,932	2,134	2,450	2,480	2,355	27	1,081,561
08	84,287	94,156	98,523	92,322	5,545	6,479	6,474	6,166	23	1,966,885
09	30,150	34,861	33,969	32,993	1,895	2,154	2,169	2,073	23	895,030
10	62,213	72,909	69,948	68,357	3,207	3,704	3,645	3,518	32	1,303,688
Min	18,483	21,745	21,133	20,454	1,464	1,656	1,662	1,594	23	812,789
Mean	50,555	58,077	57,746	55,459	2,907	3,328	3,359	3,198	32	1,192,684
Median	35,800	40,759	41,895	39,485	2,320	2,666	2,707	2,564	32	1,147,441
Max	128,243	149,091	143,728	140,354	5,545	6,479	6,474	6,166	45	1,966,885

Table 20. Summary of Incremental Effect of Branching Priority Constraints

	Speedup			LP Upper Bound Improvement
	Ave. CPU Time	Ticks	Ave. Real Time	
n = 10				
Min	0.54	1.00	0.86	0.00%
Mean	0.80	1.00	**1.04**	0.00%
Median	0.73	1.00	**1.05**	0.00%
Max	**1.19**	1.00	**1.20**	0.00%
n = 20				
Min	0.98	**1.22**	**1.16**	0.00%
Mean	**2.28**	**1.76**	**1.76**	0.00%
Median	**1.72**	**1.41**	**1.31**	0.00%
Max	**5.30**	**3.13**	**3.67**	0.00%
n = 30				
Min	**1.09**	0.98	0.94	0.00%
Mean	**2.22**	**1.58**	**1.80**	0.00%
Median	**1.65**	**1.30**	**1.41**	0.00%
Max	**7.24**	**2.75**	**4.30**	0.00%

Table 21. CI and GCI of Branching Priority Constraints

n	CPU	Ticks	Real Time	CI	GCI
10	0.75	1.00	**1.05**	0.95	**1.5264**
20	**1.86**	**1.50**	**1.42**	**1.57**	
30	**1.81**	**1.36**	**1.51**	**1.54**	

Conclusion: After applying technique 5, Branching Priority, the grand composite index of speedups (GCI) was 1.5264, which means that using the branching rule was an improvement over solving the incumbent model with CPLEX's default settings. Therefore, we adopted the branching priority on the *x* variables. Hereinafter we refer to the process of solving the incumbent model with the branching rule as the "incumbent model".

4.8 Technique 6：Lifted MTZ

Desrochers and Laporte (1991) proved that the MTZ subtour elimination constraints

$$s_i - s_j + (n+1)x_{ij} \leq n \quad (i,\ j) \in A \tag{2l}$$

can be strengthened by lifting them to

$$s_i - s_j + (n-1)\,x_{ij} + (n-3)\,x_{ji} \leq n-2 \qquad \{(i,j) \in A:\ i \neq 1,\ j \neq n\} \tag{2l'}$$

Tables 22, 23, and 24 summarize the effect of lifting the MTZ constraints on the incumbent model.

Table 22.a Test Results of Incremental Effect of Lifted MTZ for *n*= 10.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	1.02	1.05	1.04	1.04	0.60	0.57	0.54	0.57	2.00	258.33
02	1.82	5.84	4.40	4.02	0.54	0.51	0.64	0.56	2.00	81.92
03	1.44	2.29	3.19	2.31	0.55	0.66	0.61	0.61	2.57	183.24
04	2.59	1.99	2.16	2.25	0.69	0.69	0.68	0.69	4.04	207.22
05	0.68	0.84	0.73	0.75	0.30	0.34	0.33	0.32	5.34	106.03
06	4.99	2.87	2.30	3.39	0.68	0.75	0.61	0.68	6.32	196.30
07	5.03	2.64	6.07	4.58	0.82	0.83	0.92	0.86	3.09	279.09
08	1.63	1.94	1.72	1.76	0.63	0.73	0.65	0.67	3.29	230.68
09	1.11	1.14	0.87	1.04	0.48	0.50	0.42	0.47	4.36	144.46
10	1.73	1.60	1.80	1.71	0.26	0.27	0.26	0.26	6.58	43.37
Min	0.68	0.84	0.73	0.75	0.26	0.27	0.26	0.26	2.00	43.37
Mean	2.20	2.22	2.43	2.28	0.56	0.59	0.57	0.57	3.96	173.06
Median	1.68	1.96	1.98	2.00	0.58	0.62	0.61	0.59	3.67	189.77
Max	5.03	5.84	6.07	4.58	0.82	0.83	0.92	0.86	6.58	279.09

Table 22.b Test Results of Incremental Effect of Lifted MTZ for $n = 20$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	865	827	875	856	59	56	59	58	15	30,043
02	812	768	800	793	50	47	49	49	14	22,898
03	1,028	987	1,042	1,019	64	61	65	63	14	30,069
04	996	968	1,005	990	97	94	96	96	14	59,331
05	87	88	93	89	31	30	31	30	18	22,892
06	950	923	980	951	105	104	109	106	13	71,063
07	1,038	998	1,026	1,021	62	60	61	61	13	27,165
08	740	705	734	726	42	39	41	41	10	16,731
09	326	321	337	328	43	41	42	42	14	29,293
10	406	390	397	398	76	74	74	75	19	57,101
Min	497	88	93	89	31	30	31	30	10	16,731
Mean	497	698	729	717	63	61	63	62	14	36,659
Median	497	798	838	825	61	58	60	60	14	29,668
Max	497	998	1,042	1,021	105	104	109	106	19	71,063

Table 22.c Test Results of Incremental Effect of Lifted MTZ for $n = 30$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	109,652	143,250	114,980	122,627	7,342	8,538	7,606	7,829	28	2,434,610
02	80,272	136,244	78,560	98,359	4,311	6,188	4,200	4,900	27	1,745,495
03	27,701	45,590	27,481	33,591	2,636	3,401	2,625	2,888	25	1,436,603
04	23,347	34,822	22,780	26,983	1,953	2,385	1,917	2,085	32	1,061,262
05	10,405	13,412	10,256	11,358	1,924	1,956	1,871	1,917	20	1,380,103
06	38,604	41,558	36,964	39,042	2,895	2,890	2,804	2,863	34	1,250,096
07	27,540	33,841	27,711	29,697	2,354	2,544	2,321	2,406	22	1,340,991
08	82,520	87,352	83,047	84,306	5,915	5,958	6,051	5,975	18	2,219,737
09	17,824	19,097	17,794	18,238	1,704	1,685	1,719	1,702	19	1,016,520
10	24,576	25,996	24,567	25,046	2,399	2,381	2,373	2,384	26	1,424,402
Min	10,405	13,412	10,256	11,358	1,704	1,685	1,719	1,702	18	1,016,520
Mean	44,244	58,116	44,414	48,925	3,343	3,792	3,349	3,495	25	1,530,982
Median	27,620	38,190	27,596	31,644	2,518	2,717	2,499	2,635	25	1,402,253
Max	109,652	143,250	114,980	122,627	7,342	8,538	7,606	7,829	34	2,434,610

Table 23. Summary of Incremental Effect of Lifted MTZ Constraints

	Speedup			LP Upper Bound Improvement
	Ave. CPU Time	Ticks	Ave. Real Time	
n = 10				
Min	0.92	0.96	0.81	0.00%
Mean	**2.21**	**1.09**	**1.13**	16.70%
Median	**1.76**	**1.01**	**1.14**	16.98%
Max	**5.03**	**1.66**	**1.34**	33.13%
n = 20				
Min	0.82	0.59	0.72	15.05%
Mean	**1.05**	0.96	0.99	25.49%
Median	0.96	1.00	1.00	26.20%
Max	**1.41**	**1.10**	**1.12**	36.57%
n = 30				
Min	0.55	0.58	0.54	14.32%
Mean	**1.35**	0.79	0.97	20.42%
Median	**1.18**	0.83	0.97	20.84%
Max	**2.73**	0.92	**1.48**	29.41%

Table 24. CI and GCI of Lifted MTZ Constraints

n	CPU	Ticks	Real Time	CI	GCI
10	**1.88**	**1.04**	**1.14**	**1.30**	0.9991
20	0.98	0.99	0.99	0.99	
30	**1.22**	0.82	0.97	0.98	

Conclusion: After applying technique 6, Lifted MTZ, the grand composite index of speedups (GCI) was 0.9991, which means that the incumbent model was solved faster. Therefore, we did not adopt technique 6.

4.9 Technique 7：MTZ upper bound

In the original MTZ subtour elimination constraint by Miller et al. (1960), there is no upper limit for the sequence variable s_i. As result any given tour has essentially an infinite number of representations in terms of the sequence variables. This type of symmetry can needlessly slow down the branch-and-bound process by causing it "to explore and eliminate such alternative symmetric solutions" (Sherali and Smith (2001)). Desrochers and Laporte (1991) proved that constraining $1 \leq s_i \leq n-1$ ensures that there is only one representation of any given feasible tour. The effect of including upper bounds on the sequence variables are summarized in Tables 25, 26, and 27

Table 25.a Test Results of Incremental Effect of MTZ Upper Bound for $n = 10$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	3.28	8.63	3.44	5.12	0.62	0.89	0.70	0.74	2.00	266.90
02	3.63	1.51	2.48	2.54	0.48	0.41	0.49	0.46	2.00	128.12
03	3.39	4.95	4.36	4.23	0.53	0.84	0.91	0.76	2.94	157.75
04	2.21	2.72	3.20	2.71	0.79	0.85	0.61	0.75	5.01	189.11
05	5.00	1.54	2.45	3.00	0.53	0.38	0.44	0.45	6.50	153.93
06	3.80	1.47	2.27	2.52	0.51	0.42	0.49	0.47	8.60	174.29
07	6.14	2.65	1.95	3.58	0.81	0.78	0.66	0.75	3.47	262.73
08	4.92	5.21	3.05	4.39	0.88	1.04	0.90	0.94	3.92	401.28
09	0.94	0.96	1.13	1.01	0.43	0.42	0.49	0.45	6.29	166.05
10	2.50	1.89	1.84	2.08	0.47	0.33	0.26	0.35	9.84	47.70
Min	0.94	0.96	1.13	1.01	0.43	0.33	0.26	0.35	2.00	47.70
Mean	3.58	3.15	2.62	3.12	0.61	0.64	0.60	0.61	5.06	194.79
Median	3.51	2.27	2.46	2.85	0.53	0.60	0.55	0.61	4.46	170.17
Max	6.14	8.63	4.36	5.12	0.88	1.04	0.91	0.94	9.84	401.28

Table 25.b Test Results of Incremental Effect of MTZ Upper Bound for n =2 0.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	1,013	786	838	879	85	67	70	74	18	39,289
02	848	709	754	770	51	44	45	47	19	21,355
03	1,102	891	939	977	76	65	67	69	22	34,606
04	1,040	982	1,038	1,020	107	98	103	103	20	65,589
05	185	168	184	179	31	27	29	29	24	19,799
06	1,409	1,489	1,575	1,491	119	114	119	117	15	67,825
07	1,098	906	969	991	68	54	58	60	18	24,179
08	778	684	736	733	63	54	56	58	14	27,443
09	240	205	219	221	53	42	44	46	19	28,029
10	481	358	384	408	70	51	53	58	24	33,074
Min	497	168	184	179	31	27	29	29	14	19,799
Mean	497	718	764	767	72	62	65	66	19	36,119
Median	497	747	796	825	69	54	57	59	19	30,551
Max	497	1,489	1,575	1,491	119	114	119	117	24	67,825

Table 25.c Test Results of Incremental Effect of MTZ Upper Bound for $n = 30$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	188,264	111,239	119,017	139,507	20,113	8,741	9,860	12,904	35	3,053,978
02	166,376	119,388	126,284	137,349	10,236	5,071	5,350	6,885	34	1,568,098
03	49,362	26,448	27,838	34,549	5,159	2,361	2,485	3,335	32	1,372,190
04	25,516	17,155	18,245	20,305	2,999	1,395	1,467	1,953	45	752,547
05	56,931	43,182	44,965	48,359	5,286	2,561	2,627	3,491	24	1,056,220
06	93,815	44,295	46,468	61,526	6,291	2,620	2,753	3,888	41	1,138,944
07	71,092	50,987	53,842	58,640	6,691	2,899	3,056	4,216	27	1,249,116
08	211,113	107,294	111,801	143,403	13,785	5,895	6,167	8,616	23	1,807,981
09	29,726	17,404	18,199	21,776	3,133	1,513	1,584	2,077	23	823,056
10	43,140	35,612	37,340	38,697	2,859	2,434	2,538	2,610	32	1,176,301
Min	25,516	17,155	18,199	20,305	2,859	1,395	1,467	1,953	23	752,547
Mean	93,533	57,300	60,400	70,411	7,655	3,549	3,789	4,998	32	1,399,843
Median	64,012	43,738	45,716	53,500	5,788	2,591	2,690	3,690	32	1,212,708
Max	211,113	119,388	126,284	143,403	20,113	8,741	9,860	12,904	45	3,053,978

Table 26. Summary of Incremental Effect of MTZ upper bound Constraints

	Speedup			LP Upper Bound Improvement
	Ave. CPU Time	Ticks	Ave. Real Time	
$n = 10$				
Min	0.26	0.57	0.60	0.00%
Mean	**1.73**	1.00	**1.10**	0.00%
Median	**1.40**	0.96	0.96	0.00%
Max	**5.20**	**1.87**	**1.92**	0.00%
$n = 20$				
Min	0.54	0.67	0.73	0.00%
Mean	0.99	0.96	0.93	0.00%
Median	0.98	0.99	0.95	0.00%
Max	**1.31**	**1.20**	**1.05**	0.00%
$n = 30$				
Min	0.44	0.46	0.33	0.00%
Mean	0.93	0.91	0.74	0.00%
Median	0.83	0.94	0.76	0.00%
Max	**1.77**	**1.11**	**1.35**	0.00%

Table 27. CI and GCI of MTZ upper bound Constraints

n	CPU	Ticks	Real Time	CI	GCI
10	**1.49**	0.97	0.99	**1.12**	0.9096
20	0.98	0.98	0.94	0.97	
30	0.85	0.93	0.76	0.84	

Conclusion: After applying technique 7, MTZ upper bound, the grand composite index of speedups (GCI) was 0.9096 indicating that it was more efficient to solve the incumbent model. Therefore, we did not adopt the upper bound constraints for the MTZ sequence variables.

4.10 Technique 8：Cover Cuts

Fischetti et al. (1998) found that cover cuts on sets of arcs whose total length is more than the maximum route distance *D* were effective for solving the Orienteering Problem, which is a special case of the BPMP. They also proposed solving a knapsack problem to determine if there is a set of arcs S that violates the cover cut $\sum_{(i,j)\in S} x_{ij} > |S| - 1$ in the LP relaxation. We applied this technique iteratively to the BPMP adding violated cover cuts as necessary until no additional cover cuts are found at which point we solve the MIP. The results are given in Tables 28, 29, and 30.

Table 28.a Test Results of Incremental Effect of Cover Cuts for *n* =10.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	3.91	2.16	1.92	2.66	0.89	0.841569	0.85	0.86	2.00	255.37
02	4.91	5.89	4.18	4.99	0.90	0.77	1.04	0.91	2.00	91.28
03	4.86	4.74	5.46	5.02	0.91	0.71	0.84	0.82	2.94	188.63
04	3.23	2.63	5.25	3.70	0.73	1.03	1.07	0.94	5.01	206.10
05	3.84	2.29	8.14	4.76	0.62	0.77	0.93	0.77	6.50	111.96
06	5.66	5.66	12.58	7.97	1.02	1.59	1.49	1.37	8.60	333.89
07	3.64	3.12	8.29	5.02	1.12	1.47	1.44	1.34	3.47	279.35
08	2.70	2.79	3.97	3.15	0.91	0.83	0.94	0.89	3.92	235.57
09	3.26	3.94	3.42	3.54	0.80	0.83	0.79	0.81	6.29	165.98
10	1.88	0.76	1.13	1.26	0.40	0.33	0.38	0.37	9.84	43.82
Min	2.70	0.76	1.13	1.26	0.62	0.33	0.38	0.37	2.00	43.82
Mean	4.00	3.40	5.43	4.21	0.88	0.92	0.98	0.91	5.06	191.20
Median	3.84	2.96	4.72	4.23	0.90	0.83	0.93	0.88	4.46	197.37
Max	5.66	5.89	12.58	7.97	1.12	1.59	1.49	1.37	9.84	333.89

Table 28.b Test Results of Incremental Effect of Cover Cuts for $n = 20$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	1,201	803	750	918	81	57	55	65	18	28,721
02	1,415	794	737	982	80	48	48	59	19	20,938
03	1,331	985	887	1,068	97	73	68	79	22	33,802
04	2,023	1,704	1,513	1,747	155	130	118	134	20	66,270
05	52,811	52,381	50,306	51,833	7,399	7,244	7,082	7,242	24	24,233
06	1,590	1,285	973	1,283	139	90	77	102	15	46,357
07	1,301	974	884	1,053	82	63	59	68	18	26,624
08	792	733	688	738	48	47	45	47	14	19,324
09	375	345	279	333	65	62	57	61	19	38,576
10	1,000	605	516	707	99	60	55	71	24	33,488
Min	375	345	279	333	48	47	45	47	14	19,324
Mean	6,384	6,061	5,753	6,066	825	787	767	793	19	33,833
Median	1,316	888	817	1,017	89	62	58	70	19	31,105
Max	52,811	52,381	50,306	51,833	7,399	7,244	7,082	7,242	24	66,270

Table 28.c Test Results of Incremental Effect of Cover Cuts for $n = 30$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	163,814	110,014	100,087	124,638	11,210	7,650	6,978	8,613	35	2,471,321
02	156,964	109,168	94,901	120,344	8,054	4,697	4,206	5,653	34	1,432,326
03	53,951	37,454	32,678	41,361	4,084	2,793	2,516	3,131	32	1,246,448
04	29,781	20,245	17,227	22,418	2,573	1,816	1,639	2,009	45	968,206
05	36,947	29,370	25,515	30,611	3,062	1,996	1,793	2,283	24	936,560
06	52,662	35,395	34,936	40,998	3,433	2,325	2,235	2,665	41	913,807
07	35,181	25,640	22,331	27,717	3,549	2,375	2,200	2,708	27	1,268,084
08	111,663	76,245	72,748	86,885	7,185	4,888	4,620	5,564	23	1,644,209
09	31,972	30,566	26,140	29,559	2,716	2,022	1,801	2,179	23	874,020
10	48,201	41,012	35,332	41,515	3,769	2,693	2,456	2,973	32	1,137,796
Min	29,781	20,245	17,227	22,418	2,573	1,816	1,639	2,009	23	874,020
Mean	72,114	51,511	46,189	56,605	4,964	3,326	3,044	3,778	32	1,289,278
Median	50,431	36,425	33,807	41,179	3,659	2,534	2,345	2,840	32	1,192,122
Max	163,814	110,014	100,087	124,638	11,210	7,650	6,978	8,613	45	2,471,321

Table 29. Summary of Incremental Effect of Cover Cuts Constraints

	Speedup			LP Upper Bound Improvement
	Ave. CPU Time	Ticks	Ave. Real Time	
n = 10				
Min	0.16	0.69	0.42	0.00%
Mean	**1.13**	0.96	0.71	0.00%
Median	**1.22**	0.98	0.75	0.00%
Max	**1.67**	**1.06**	0.87	0.00%
n = 20				
Min	0.00	0.83	0.00	0.00%
Mean	0.74	**1.01**	0.76	0.00%
Median	0.81	0.97	0.78	0.00%
Max	0.96	**1.57**	**1.03**	0.00%
n = 30				
Min	0.54	0.57	0.50	0.00%
Mean	**1.02**	0.95	0.88	0.00%
Median	**1.04**	0.96	0.88	0.00%
Max	**1.65**	**1.20**	**1.18**	0.00%

Table 30. CI and GCI of Cover Cuts

n	CPU	Ticks	Real Time	CI	GCI
10	**1.20**	0.97	0.74	0.95	0.9069
20	0.79	0.98	0.77	0.85	
30	**1.04**	0.96	0.88	0.95	

Conclusion: After applying technique 8, Cover Cuts, the grand composite index of speedups (GCI) is 0.9069 indicating that it was more efficient to solve the incumbent model. Therefore, we did not adopt the technique of cover cuts.

4.11 Technique 9：Pairwise Demand Cuts

Pairwise demand cuts state that pairs of delivery requests from the same node whose total weight exceeds the vehicle's capacity are mutually exclusive.

$$y_{ki} + y_{kj} \leq 1 \qquad \forall\{(k,i),(k,j) \in A: i \neq j, w_{ki} + w_{kj} > Q\} \quad (2q)$$
$$y_{ik} + y_{jk} \leq 1 \qquad \forall\{(i,k),(j,k) \in A: i \neq j, w_{ik} + w_{jk} > Q\} \quad (2r)$$

The above are valid inequalities that are satisfied by any feasible solution to the MIP formulation. In our preliminary tests, we observed that these cuts were not present in every possible case. Therefore, instead of adding them as additional constraints to the whole node-arc model, we adopt a simple scheme to add them as necessary. That is, we check for violated pairwise demand cuts of the corresponding LP relaxation, add any violated cuts found to the model, and solve the LP again. This process is repeated until no more cuts are found in the LP relaxation problem, at which point we restore the integrality constraints and solve the MIP. In this way, we can use a minimal number

of pairwise demand cuts. We denote the set of node pairs for which pairwise demand cuts are added by *B*. Tables 31, 32, and 33 give the results obtained from applying this technique to the incumbent model.

Table 31.a Test Results of Incremental Effect of Pairwise Demand Cuts for $n = 10$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	1.91	1.60	1.71	1.74	1.38	1.02689	1.07	1.16	2.00	245.31
02	3.96	4.22	3.34	3.84	0.79	0.95	0.73	0.82	2.00	91.28
03	3.20	2.73	2.82	2.92	1.00	0.89	0.93	0.94	2.94	189.06
04	2.69	1.81	3.07	2.52	0.93	0.88	0.83	0.88	4.65	197.43
05	1.07	1.31	1.30	1.23	0.45	0.56	0.56	0.52	6.50	111.96
06	2.06	2.38	2.39	2.28	1.05	1.16	1.20	1.14	8.48	343.63
07	5.34	4.93	2.93	4.40	1.03	1.28	1.00	1.10	3.47	276.50
08	1.75	3.13	4.57	3.15	0.69	1.14	0.95	0.93	3.88	242.67
09	3.42	2.92	3.32	3.22	0.84	0.98	0.96	0.93	6.29	166.29
10	1.02	0.82	0.82	0.89	0.61	0.53	0.56	0.57	9.80	50.95
Min	1.07	0.82	0.82	0.89	0.45	0.53	0.56	0.52	2.00	50.95
Mean	2.82	2.59	2.63	2.62	0.91	0.94	0.88	0.90	5.00	191.51
Median	2.69	2.56	2.87	2.72	0.93	0.97	0.94	0.93	4.27	193.25
Max	5.34	4.93	4.57	4.40	1.38	1.28	1.20	1.16	9.80	343.63

Table 31.b Test Results of Incremental Effect of Pairwise Demand Cuts for $n = 20$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	1,473	805	873	1,050	94	62	63	73	18	27,321
02	1,212	736	763	903	75	50	52	59	19	22,714
03	1,161	978	1,055	1,065	77	68	71	72	22	28,821
04	1,237	1,170	1,232	1,213	118	117	124	120	20	59,825
05	182	179	183	181	44	43	44	44	24	26,643
06	1,265	1,148	1,234	1,216	120	117	123	120	15	73,393
07	1,555	903	974	1,144	95	64	65	74	18	26,859
08	1,345	855	918	1,039	97	69	72	79	14	26,292
09	496	306	322	374	90	69	72	77	19	38,632
10	390	340	361	364	56	51	53	53	24	30,898
Min	182	179	183	181	44	43	44	44	14	22,714
Mean	1,031	742	791	855	87	71	74	77	19	36,140
Median	1,225	830	895	1,045	92	66	68	74	19	28,071
Max	1,555	1,170	1,234	1,216	120	117	124	120	24	73,393

Table 31.c Test Results of Incremental Effect of Pairwise Demand Cuts for n =30.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	59,409	57,242	61,217	59,289	3,641	3,516	3,732	3,630	35	1,256,904
02	33,662	31,838	28,706	31,402	2,414	2,347	2,537	2,433	33	1,050,958
03	45,848	44,190	47,043	45,694	2,839	2,794	2,950	2,861	32	1,202,606
04	31,271	29,742	34,339	31,784	2,232	2,129	2,431	2,264	45	826,145
05	42,705	40,546	42,742	41,998	2,553	2,482	2,649	2,561	23	918,572
06	42,163	43,748	45,662	43,858	2,715	2,783	2,902	2,800	41	1,082,116
07	45,982	46,030	48,503	46,838	3,108	3,160	3,329	3,199	27	1,307,731
08	79,578	79,015	82,460	80,351	5,164	5,177	5,377	5,239	23	1,775,436
09	25,491	24,582	25,867	25,313	1,788	1,750	1,853	1,797	23	829,883
10	39,401	38,339	40,187	39,309	2,604	2,539	2,656	2,600	32	1,137,796
Min	25,491	24,582	25,867	25,313	1,788	1,750	1,853	1,797	23	826,145
Mean	44,551	43,527	45,673	44,584	2,906	2,868	3,042	2,938	32	1,138,815
Median	42,434	42,147	44,202	42,928	2,659	2,661	2,779	2,700	32	1,109,956
Max	79,578	79,015	82,460	80,351	5,164	5,177	5,377	5,239	45	1,775,436

Table 32. Summary of Incremental Effect of Pairwise Demand Cuts

	Speedup			LP Upper Bound Improvement
	Ave. CPU Time	Ticks	Ave. Real Time	
n = 10				
Min	0.63	0.67	0.38	0.00%
Mean	**1.66**	0.95	0.71	1.00%
Median	**1.65**	0.98	0.75	0.00%
Max	**2.35**	**1.06**	0.88	7.06%
n = 20				
Min	0.66	0.70	0.54	0.00%
Mean	0.87	0.95	0.78	0.59%
Median	0.81	0.97	0.81	0.29%
Max	**1.47**	**1.08**	1.00	1.83%
n = 30				
Min	0.51	0.80	0.64	0.00%
Mean	**1.34**	**1.05**	**1.09**	0.32%
Median	**1.02**	**1.04**	**1.06**	0.22%
Max	**4.47**	**1.47**	**2.26**	1.09%

Table 33. CI and GCI of Pairwise Demand Cuts

n	CPU	Ticks	Real Time	CI	GCI
10	**1.65**	0.97	0.74	**1.07**	0.9853
20	0.83	0.97	0.80	0.87	
30	**1.11**	**1.05**	**1.07**	**1.08**	

Conclusion: After applying technique 9, Pairwise Demand Cuts, the grand composite index of speedups (GCI) was 0.9853, which means that the incumbent model was solved faster. Therefore, we did not adopt technique 9.

4.12 Summary of Enhanced Node-Arc Model and Results for 40-Node Instance

We conclude this section by restating the enhanced model and applying it to the 40-node problem instances.

Enhanced Node-Arc Model

Maximize $p\left[\sum_{(k,\,l)\in E} d_{kl} w_{kl} y_{kl}\right] - c\sum_{(i,\,j)\in E} \theta_{ij} d_{ij} - cv\sum_{(i,\,j)\in E} d_{ij} x_{ij}$

Subject to:

$\sum_{j\in V} z_{kl,kj} = y_{kl} \quad (k,\, l) \in A$

$\sum_{i\in V} z_{kl,il} = y_{kl} \quad (k,\, l) \in A$

$\sum_{i\in V,\,(i,\,a)\in A} z_{kl,ia} = \sum_{j\in V,\,(a,\,j)\in A} z_{kl,aj} \quad (k,\, l) \in A,\, a \in V\backslash\{k, l\}$

$\sum_{(1,j)\in A} x_{1,j} = 1$

$\sum_{(i,n)\in A} x_{i,n} = 1$

$\sum_{i\in V\backslash\{k,\,n\}} x_{ik} = \sum_{j\in V\backslash\{1,\,k\}} x_{kj} \quad k \in V\backslash\{1, n\}$

$\sum_{(i,\,j)\in A} d_{ij} x_{ij} \le D$

$\theta_{ij} = \sum_{(k,\,l)\in A} w_{kl} z_{kl,ij} \quad (i,\, j) \in A$

$\theta_{ij} \le Q x_{ij} \quad (i,\, j) \in A$ (Conditional Arc Flow)

$s_i - s_j + (n+1)x_{ij} \le n \quad (i,\, j) \in A$

$x_{ij} \in \{0,\, 1\} \quad (i,\, j) \in A$

$y_{kl} \in \{0,\, 1\} \quad (k,\, l) \in A$

$z_{kl,ij} \in \{0,\ 1\}\ \ (k,\ l) \in A, (i,\ j) \in A$

We conclude this section with results from applying the enhanced node-arc model to 40-node BPMP instances. From Table 34, we can see that the median and average real time for 40-node instances was 56,428 seconds (15.7 hours) and 329,773 seconds (91.6 hours), which is probably impractical for real-world application.

Table 34a. CPU Times for Enhanced Node-Arc Model for *n* = 40.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time
01	1,642,420	1,609,370	1,624,790	1,625,527
02	5,311,600	5,398,720	5,554,910	5,421,743
03	2,279,190	2,279,210	2,294,180	2,284,193
04	440,044	432,906	439,291	437,414
05	504,116	501,716	506,084	503,972
06	895,623	893,713	901,347	896,894
07	1,322,480	1,309,680	1,320,570	1,317,577
08	1,124,880	1,095,630	1,108,450	1,109,653
09	334,320	331,986	335,769	334,025
10	50,837,400	54,439,600	52,230,100	52,502,367
Min	334,320	331,986	335,769	334,025
Mean	6,469,207	6,829,253	6,631,549	6,643,337
Median	1,223,680	1,202,655	1,214,510	1,213,615
Max	50,837,400	54,439,600	52,230,100	52,502,367

Table 34b. Real Times for Enhanced Node-Arc Model for n = 40.

Instance	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time
01	76,067	74,799	75,979	75,615
02	190,590	193,773	200,240	194,868
03	139,011	137,874	139,658	138,847
04	23,039	22,795	23,035	22,956
05	30,046	29,894	30,123	30,021
06	57,037	57,189	57,075	57,100
07	56,135	55,357	55,778	55,756
08	52,482	50,920	51,489	51,630
09	18,474	18,300	18,464	18,413
10	2,557,613	2,762,254	2,637,688	2,652,518
Min	18,474	18,300	18,464	18,413
Mean	320,049	340,315	328,953	329,773
Median	56,586	56,273	56,427	56,428
Max	2,557,613	2,762,254	2,637,688	2,652,518

Table 34c. LP Upper Bounds and Ticks for Enhanced Node-Arc Model for n = 40.

Instance	LP Upper Bound	Ticks
01	49	18,661,092
02	57	38,797,945
03	54	28,592,204
04	37	8,250,144
05	66	9,998,664
06	52	13,904,434
07	41	15,569,350
08	38	16,177,156
09	55	6,601,682
10	61	463,811,772
Min	37	6,601,682
Mean	51	62,036,444
Median	53	15,873,253
Max	66	463,811,772

5. ENHANCING THE TRIPLES FORMULATION

5.1 Initial Incumbent Formulation

In this section we give the results for our initial incumbent, the original triples model. The model was solved three times for each problem instance. The results are shown in Tables 35.

Table 35.a Test Results of Original Triples Model for $n = 10$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	3.15	1.91	2.69	2.58	0.68	0.62	0.74	0.68	2.00000	39.06
02	3.14	4.70	0.71	2.85	0.56	0.47	0.39	0.47	2.00000	34.44
03	2.69	2.82	3.61	3.04	0.42	0.48	0.48	0.46	2.00000	49.68
04	4.55	5.58	2.73	4.29	0.97	0.87	0.90	0.91	2.00000	97.23
05	2.53	3.66	2.61	2.93	0.54	0.59	0.54	0.56	2.00000	27.97
06	2.41	2.07	4.43	2.97	0.41	0.45	0.54	0.47	2.00000	66.53
07	2.80	3.69	2.69	3.06	0.76	0.49	0.57	0.61	2.00000	52.56
08	1.52	1.09	3.05	1.89	0.37	0.39	0.40	0.39	2.00000	62.65
09	5.24	2.48	1.87	3.20	0.47	0.45	0.51	0.48	2.00000	39.59
10	1.13	1.23	0.58	0.98	0.10	0.11	0.15	0.12	2.00000	10.63
Min	1.13	1.09	0.58	0.98	0.10	0.11	0.15	0.12	2.00000	10.63
Mean	2.92	2.92	2.50	2.78	0.53	0.49	0.52	0.51	2.00000	48.03
Median	2.75	2.65	2.69	2.95	0.51	0.48	0.53	0.48	2.00000	44.64
Max	5.24	5.58	4.43	4.29	0.97	0.87	0.90	0.91	2.00000	97.23

Table 35.b Test Results of Original Triples Model for $n = 20$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	11.82	12.36	11.14	11.78	3.31	3.15	3.25	3.24	2.00024	1,382.92
02	5.89	4.82	6.39	5.70	2.43	2.28	2.41	2.37	2.00012	705.12
03	17.17	15.90	16.73	16.60	3.89	3.80	3.80	3.83	2.00060	1,776.09
04	44.74	44.71	46.30	45.25	5.35	5.23	5.40	5.33	2.00063	2,600.35
05	14.57	14.34	14.59	14.50	3.03	2.89	2.97	2.96	2.00000	1,441.45
06	308.56	328.86	306.95	314.79	20.93	21.64	20.48	21.02	2.00180	13,753.46
07	9.08	9.39	9.29	9.26	3.06	2.99	3.09	3.05	2.00024	1,136.33
08	9.26	8.36	8.63	8.75	3.57	3.22	3.31	3.37	2.00000	1,313.38
09	70.57	72.79	67.04	70.14	5.49	5.60	5.37	5.49	2.00204	2,809.86
10	25.38	27.30	24.66	25.78	3.56	3.35	3.40	3.44	2.00000	1,572.45
Min	5.89	4.82	6.39	5.70	2.43	2.28	2.41	2.37	2.00000	705.12
Mean	51.70	53.88	51.17	52.25	5.46	5.42	5.35	5.41	2.00057	2,849.14
Median	15.87	15.12	15.66	15.55	3.57	3.29	3.36	3.40	2.00024	1,506.95
Max	308.56	328.86	306.95	314.79	20.93	21.64	20.48	21.02	2.00204	13,753.46

Table 35.c Test Results of Original Triples Model for $n = 30$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	2,007	2,187	2,057	2,084	106	115	107	109	2.00000	57,767
02	6,624	6,986	6,783	6,798	316	332	322	323	2.00084	190,088
03	1,552	1,657	1,626	1,611	106	112	109	109	2.00076	64,027
04	477	492	477	482	78	80	78	79	2.00103	60,791
05	2,288	2,442	2,368	2,366	121	127	122	123	2.00104	63,157
06	30	30	31	30	7	6	6	6	2.00146	4,318
07	2,120	2,286	2,236	2,214	119	127	123	123	2.00219	65,146
08	2,275	2,416	2,324	2,338	129	135	129	131	2.00124	71,547
09	6,352	6,773	6,584	6,570	398	424	403	408	2.00151	259,044
10	10,181	10,726	10,533	10,480	561	587	580	576	2.00116	248,936
Min	30	30	31	30	7	6	6	6	2.00000	4,318
Mean	3,390	3,600	3,502	3,497	194	205	198	199	2.00112	108,482
Median	2,198	2,351	2,280	2,276	120	127	123	123	2.00110	64,586
Max	10,181	10,726	10,533	10,480	561	587	580	576	2.00219	259,044

Table 35.d Test Results of Original Triples Model for $n = 40$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	9,159	9,485	9,424	9,356	505	522	510	512	2.00002	223,051
02	230,646	233,806	233,457	232,636	12,633	12,967	12,762	12,787	2.00002	2,231,392
03	9,941	10,380	10,221	10,181	464	485	474	474	2.00003	200,479
04	4,924	5,099	5,097	5,040	357	384	363	368	2.00004	216,198
05	30,709	31,393	31,534	31,212	1,532	1,578	1,551	1,553	2.00005	583,752
06	110	115	110	112	16	17	16	17	2.00005	10,844
07	52,311	53,147	53,447	52,968	2,495	2,566	2,529	2,530	2.00002	719,292
08	117,263	120,211	120,584	119,353	5,383	5,500	5,464	5,449	2.00003	1,273,891
09	84,683	86,619	86,434	85,912	3,481	3,565	3,559	3,535	2.00003	1,006,632
10	102,631	105,095	104,735	104,154	5,021	5,141	5,088	5,083	2.00005	1,427,219
Min	110	115	110	112	16	17	16	17	2.00002	10,844
Mean	64,238	65,535	65,504	65,092	3,189	3,273	3,232	3,231	2.00003	789,275
Median	41,510	42,270	42,490	42,090	2,013	2,072	2,040	2,042	2.00003	651,522
Max	230,646	233,806	233,457	232,636	12,633	12,967	12,762	12,787	2.00005	2,231,392

Table 35.e Test Results of Original Triples Model for n = 50.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	265,863	266,362	267,515	266,580	10,802	10,868	10,891	10,853	2.00108	2,310,519
02	810,851	800,358	794,515	801,908	38,720	39,002	38,191	38,638	2.00196	6,807,569
03	678,901	680,901	670,418	676,740	28,237	28,427	27,970	28,211	2.00186	5,457,936
04	76,866	76,762	76,741	76,790	3,293	3,360	3,296	3,316	2.00281	1,059,798
05	127,561	127,923	127,392	127,625	5,659	5,767	5,719	5,715	2.00104	1,531,558
06	295,880	293,908	296,457	295,415	13,551	13,573	13,561	13,562	2.00324	2,717,566
07	598,680	599,843	598,380	598,968	25,062	25,265	25,062	25,130	2.00144	4,654,508
08	101,607	102,628	101,456	101,897	4,148	4,238	4,169	4,185	2.00153	1,390,149
09	265,103	264,168	264,372	264,548	11,312	11,400	11,337	11,350	2.00218	2,501,357
10	1,162,570	1,172,530	1,178,230	1,171,110	46,822	47,669	47,508	47,333	2.00111	8,281,539
Min	76,866	76,762	76,741	76,790	3,293	3,360	3,296	3,316	2.00104	1,059,798
Mean	438,388	438,538	437,548	438,158	18,761	18,957	18,770	18,829	2.00183	3,671,250
Median	280,872	280,135	281,986	280,998	12,432	12,487	12,449	12,456	2.00170	2,609,461
Max	1,162,570	1,172,530	1,178,230	1,171,110	46,822	47,669	47,508	47,333	2.00324	8,281,539

As shown in Tables 35.a-e, we were able to solve 10-node through 50-node instances with the original triples model. But the mean real time for 50-node instances was more than 5 hours, which is not practical in the real world. There are no speedups yet (no techniques applied yet).

5.2 Technique 1：Relax Linking Constraints

The linking constraint in section 2.3 (3d) is to force (i, k) to be an arc on the vehicle's route if variable u_{ij}^{k} is positive. However, Dong (2015) showed that model remains valid even if this constraint is relaxed. Relaxing (3d) significantly reduces the number of constraints in the triples model and consequently improves solution time as shown in Tables 36, 37, and 38.

Table 36.a Test Results of Incremental Effect of Relaxing Triples Linking Constraints for n =10.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	2.84	9.23	1.30	4.46	0.48	0.59	0.43	0.50	2.00000	52.93
02	6.57	8.40	1.53	5.50	0.57	0.61	0.41	0.53	2.00000	27.67
03	3.68	1.25	1.37	2.10	0.35	0.27	0.36	0.33	2.00000	39.63
04	1.44	5.55	1.79	2.93	0.26	0.40	0.26	0.31	2.00000	58.17
05	2.53	3.87	1.84	2.75	0.30	0.36	0.30	0.32	2.00000	35.92
06	2.23	5.48	2.63	3.45	0.26	0.45	0.36	0.36	2.00000	55.46
07	3.64	4.41	2.37	3.48	0.68	0.72	0.67	0.69	2.00000	43.87
08	4.42	2.40	1.69	2.83	0.40	0.30	0.35	0.35	2.00000	94.04
09	2.18	3.72	0.74	2.21	0.22	0.30	0.22	0.25	2.00000	32.39
10	1.10	1.30	1.69	1.36	0.11	0.11	0.13	0.12	2.00000	20.06
Min	1.10	1.25	0.74	1.36	0.11	0.11	0.13	0.12	2.00000	20.06
Mean	3.06	4.56	1.69	3.11	0.36	0.41	0.35	0.37	2.00000	46.01
Median	2.68	4.14	1.69	2.88	0.33	0.38	0.36	0.34	2.00000	41.75
Max	6.57	9.23	2.63	5.50	0.68	0.72	0.67	0.69	2.00000	94.04

Table 36.b Test Results of Incremental Effect of Relaxing Triples Linking Constraints for n = 20.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	9.83	9.74	9.73	9.77	2.27	2.29	2.24	2.27	2.00024	994.84
02	4.55	4.47	4.16	4.39	1.49	1.42	1.30	1.40	2.00012	549.77
03	25.88	26.24	25.05	25.72	5.51	5.56	5.30	5.46	2.00060	3,354.07
04	26.24	25.26	25.82	25.77	3.34	3.15	3.27	3.25	2.00063	1,374.97
05	11.13	11.48	11.18	11.26	2.54	2.64	2.52	2.57	2.00000	1,161.77
06	124.53	124.19	124.53	124.42	10.05	9.84	10.00	9.96	2.00180	6,194.38
07	6.92	6.74	6.85	6.83	2.04	1.93	2.03	2.00	2.00024	807.62
08	9.32	9.46	8.75	9.18	2.10	2.10	1.97	2.06	2.00000	834.06
09	31.08	30.91	31.42	31.14	3.45	3.40	3.50	3.45	2.00204	1,646.54
10	13.86	14.25	14.22	14.11	3.91	3.46	3.79	3.72	2.00000	1,216.68
Min	4.55	4.47	4.16	4.39	1.49	1.42	1.30	1.40	2.00000	549.77
Mean	26.33	26.27	26.17	26.26	3.67	3.58	3.59	3.61	2.00057	1,813.47
Median	12.49	12.87	12.70	12.69	2.94	2.90	2.90	2.91	2.00024	1,189.23
Max	124.53	124.19	124.53	124.42	10.05	9.84	10.00	9.96	2.00204	6,194.38

Table 36.c Test Results of Incremental Effect of Relaxing Triples Linking Constraints for $n = 30$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	1,393	1,393	1,395	1,393	65	64	64	64	2.00000	47,098
02	761	759	757	759	50	49	49	49	2.00084	41,553
03	540	543	540	541	31	32	31	31	2.00076	23,678
04	53	53	52	52	10	10	10	10	2.00103	7,225
05	640	635	636	637	36	35	36	36	2.00104	25,442
06	38	38	38	38	8	8	8	8	2.00146	5,788
07	757	756	764	759	43	43	43	43	2.00219	34,530
08	674	676	674	675	39	40	40	39	2.00124	31,445
09	1,220	1,216	1,214	1,217	63	63	63	63	2.00151	49,284
10	3,246	3,257	3,262	3,255	168	168	166	167	2.00116	145,491
Min	38	38	38	38	8	8	8	8	2.00000	5,788
Mean	932	932	933	933	51	51	51	51	2.00112	41,153
Median	716	716	715	717	41	41	41	41	2.00110	32,988
Max	3,246	3,257	3,262	3,255	168	168	166	167	2.00219	145,491

Table 36.d Test Results of Incremental Effect of Relaxing Triples Linking Constraints for $n = 40$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	7,201	7,125	7,129	7,152	295	294	297	295	2.00002	188,145
02	83,958	83,701	83,890	83,850	3,065	3,057	3,041	3,054	2.00002	1,645,678
03	1,805	1,813	1,825	1,814	111	110	111	111	2.00003	91,071
04	1,238	1,226	1,235	1,233	111	110	112	111	2.00004	108,777
05	4,190	4,210	4,191	4,197	197	197	197	197	2.00005	129,188
06	110	108	107	108	15	15	15	15	2.00005	10,497
07	26,473	26,518	26,435	26,475	1,024	1,025	1,024	1,024	2.00002	629,536
08	29,710	29,807	29,837	29,784	1,223	1,223	1,227	1,224	2.00003	791,111
09	27,626	27,744	27,602	27,657	1,076	1,089	1,080	1,082	2.00003	673,787
10	45,022	45,058	44,980	45,020	1,827	1,833	1,820	1,827	2.00005	1,000,757
Min	110	108	107	108	15	15	15	15	2.00002	10,497
Mean	22,733	22,731	22,723	22,729	894	895	892	894	2.00003	526,855
Median	16,837	16,821	16,782	16,813	660	660	661	660	2.00003	408,840
Max	83,958	83,701	83,890	83,850	3,065	3,057	3,041	3,054	2.00005	1,645,678

Table 36.e Test Results of Incremental Effect of Relaxing Triples Linking Constraints for $n = 50$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	141,304	137,605	140,397	139,769	5,007	4,880	4,971	4,953	2.00108	2,058,914
02	866,252	858,891	892,573	872,572	30,469	30,362	31,393	30,741	2.00196	10,599,888
03	368,226	362,851	381,483	370,853	12,930	12,751	13,293	12,991	2.00186	4,595,967
04	67,435	65,744	67,846	67,008	2,420	2,336	2,382	2,380	2.00281	977,185
05	301,103	295,877	308,370	301,783	10,760	10,528	10,927	10,738	2.00104	3,735,936
06	388,682	385,707	405,853	393,414	13,546	13,365	14,009	13,640	2.00324	4,646,419
07	375,089	370,334	388,520	377,981	13,120	13,439	13,842	13,467	2.00144	4,614,224
08	48,649	47,324	48,210	48,061	1,811	1,771	1,809	1,797	2.00153	780,865
09	146,889	145,335	146,745	146,323	5,363	5,268	5,329	5,320	2.00218	2,140,370
10	568,862	562,414	590,824	574,033	20,187	19,526	20,516	20,076	2.00111	7,221,995
Min	48,649	47,324	48,210	48,061	1,811	1,771	1,809	1,797	2.00104	780,865
Mean	327,249	323,208	337,082	329,180	11,561	11,423	11,847	11,610	2.00183	4,137,176
Median	334,665	329,364	344,927	336,318	11,845	11,639	12,110	11,865	2.00170	4,165,951
Max	866,252	858,891	892,573	872,572	30,469	30,362	31,393	30,741	2.00324	10,599,888

Table 37. Summary of Incremental Effect of Relaxing Triples Linking Constraints

	Speedup			LP Upper Bound Improvement
	Ave. CPU Time	Ticks	Ave. Real Time	
$n = 10$				
Min	0.52	0.53	0.88	0.00%
Mean	0.97	**1.05**	**1.46**	0.00%
Median	0.87	**1.20**	**1.33**	0.00%
Max	**1.47**	**1.67**	**2.98**	0.00%
$n = 20$				
Min	0.65	0.53	0.70	0.00%
Mean	**1.51**	**1.45**	**1.44**	0.00%
Median	**1.33**	**1.40**	**1.56**	0.00%
Max	**2.53**	**2.22**	**2.11**	0.00%
$n = 30$				
Min	0.80	0.75	0.77	0.00%
Mean	**4.22**	**3.13**	**3.99**	0.00%
Median	**3.34**	**2.38**	**3.46**	0.00%
Max	**9.21**	**8.41**	**7.84**	0.00%
$n = 40$				
Min	**1.03**	**1.03**	**1.10**	0.00%
Mean	**3.37**	**1.80**	**3.55**	0.00%
Median	**2.94**	**1.46**	**3.29**	0.00%
Max	**7.44**	**4.52**	**7.87**	0.00%
$n = 50$				
Min	0.42	0.41	0.53	0.00%
Mean	**1.45**	**1.01**	**1.72**	0.00%
Median	**1.70**	**1.10**	**2.00**	0.00%
Max	**2.12**	**1.78**	**2.36**	0.00%

Table 38. CI and GCI of Relaxing Triples Linking Constraints

n	CPU	Ticks	Real Time	CI	GCI
10	0.89	**1.17**	**1.37**	**1.17**	
20	**1.37**	**1.41**	**1.53**	**1.44**	
30	**3.55**	**2.57**	**3.58**	**3.20**	**2.11**
40	**3.05**	**1.55**	**3.36**	**2.62**	
50	**1.64**	**1.09**	**1.93**	**1.55**	

Conclusion: After applying technique 1, relax linking constraints (3d), the grand composite index of speedups (GCI) was 2.11, which means, on average, the model with relax linking constraints was solved 2.11 times faster than the incumbent model. Thus, we adopted it.

5.3 Technique 2： Enforce Node-Degree

Unlike the node-arc model, the original triples model does not explicitly enforce the node-degree constraints (2h) because the MTZ subtour elimination constraints (2l) ensure that vehicle visits each node at most once in an integer solution. However, the node-degree constraints can be violated in solutions to the LP relaxation of the triples model. Tables 39, 40, and 41 summarize the effect of adding (2h) to the triples model.

Table 39.a Test Results of Incremental Effect of Enforcing Node-Degree Constraints for *n* =10.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	1.09	1.69	1.38	1.39	0.26	0.28	0.27	0.27	2.00000	42.03
02	2.29	3.35	1.49	2.37	0.30	0.35	0.23	0.29	2.00000	79.55
03	3.13	2.92	1.49	2.51	0.44	0.37	0.36	0.39	2.00000	70.13
04	0.82	1.28	5.03	2.38	0.21	0.26	0.40	0.29	2.00000	29.74
05	0.81	2.11	4.19	2.37	0.16	0.22	0.30	0.23	2.00000	35.50
06	2.50	4.27	1.86	2.87	0.53	0.59	0.53	0.55	2.00000	69.55
07	5.85	5.44	5.91	5.73	0.92	0.82	0.96	0.90	2.00000	41.21
08	2.60	1.06	0.91	1.52	0.28	0.26	0.26	0.27	2.00000	85.26
09	2.55	1.18	1.21	1.65	0.20	0.15	0.18	0.18	2.00000	29.72
10	1.29	4.14	6.02	3.82	0.11	0.26	0.33	0.23	2.00000	21.72
Min	0.81	1.06	0.91	1.39	0.11	0.15	0.18	0.18	2.00000	21.72
Mean	2.29	2.74	2.95	2.66	0.34	0.36	0.38	0.36	2.00000	50.44
Median	2.39	2.52	1.67	2.38	0.27	0.27	0.32	0.28	2.00000	41.62
Max	5.85	5.44	6.02	5.73	0.92	0.82	0.96	0.90	2.00000	85.26

Table 39.b Test Results of Incremental Effect of Enforcing Node-Degree Constraints for $n = 20$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	7.73	7.80	8.02	7.85	1.88	1.85	2.00	1.91	2.00024	806.69
02	4.53	5.11	4.13	4.59	1.83	1.93	1.62	1.79	2.00012	641.58
03	25.86	25.80	26.19	25.95	6.38	6.42	6.23	6.34	2.00060	3,539.62
04	19.64	18.96	18.67	19.09	2.21	2.14	2.01	2.12	2.00063	952.86
05	9.98	9.52	9.90	9.80	2.41	2.42	2.44	2.42	2.00000	1,080.53
06	101.83	102.91	101.94	102.22	9.59	9.63	9.71	9.64	2.00120	5,003.73
07	6.04	6.23	5.90	6.06	1.21	1.23	1.19	1.21	2.00024	955.74
08	7.56	7.63	7.85	7.68	2.43	2.42	2.47	2.44	2.00000	870.83
09	48.88	48.37	47.77	48.34	5.41	5.26	5.11	5.26	2.00120	3,190.62
10	12.69	12.16	11.99	12.28	2.76	2.71	2.85	2.77	2.00000	1,224.06
Min	4.53	5.11	4.13	4.59	1.21	1.23	1.19	1.21	2.00000	641.58
Mean	24.48	24.45	24.24	24.39	3.61	3.60	3.56	3.59	2.00042	1,826.63
Median	11.34	10.84	10.94	11.04	2.42	2.42	2.46	2.43	2.00024	1,018.14
Max	101.83	102.91	101.94	102.22	9.59	9.63	9.71	9.64	2.00120	5,003.73

Table 39.c Test Results of Incremental Effect of Enforcing Node-Degree Constraints for $n = 30$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	428	431	432	430	29	29	29	29	2.00000	22,102
02	758	758	753	756	45	45	45	45	2.00084	36,254
03	401	406	406	405	32	33	32	32	2.00076	26,200
04	84	83	83	83	23	22	22	22	2.00103	13,248
05	386	388	388	387	29	29	29	29	2.00104	19,896
06	34	34	34	34	7	7	7	7	2.00146	4,460
07	440	440	439	440	33	33	32	32	2.00201	27,300
08	476	477	476	476	30	30	30	30	2.00124	22,649
09	767	760	764	764	47	46	47	47	2.00151	37,883
10	1,188	1,192	1,206	1,195	71	72	71	71	2.00116	57,959
Min	34	34	34	34	7	7	7	7	2.00000	4,460
Mean	496	497	498	497	34	35	34	34	2.00111	26,795
Median	434	436	435	435	31	32	31	31	2.00110	24,424
Max	1,188	1,192	1,206	1,195	71	72	71	71	2.00201	57,959

Table 39.d Test Results of Incremental Effect of Enforcing Node-Degree Constraints for = 40.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	3,475	3,493	3,502	3,490	190	191	189	190	2.00002	142,912
02	58,456	58,441	58,390	58,429	2,156	2,154	2,154	2,154	2.00002	1,162,683
03	1,304	1,296	1,334	1,312	83	84	83	83	2.00003	71,249
04	850	854	865	856	85	85	86	85	2.00004	82,715
05	4,511	4,486	4,514	4,504	241	239	240	240	2.00005	179,905
06	122	116	119	119	14	14	14	14	2.00005	10,638
07	5,476	5,459	5,492	5,476	251	251	251	251	2.00002	165,771
08	22,031	21,988	22,052	22,024	917	907	912	912	2.00003	600,851
09	18,202	18,130	18,220	18,184	748	754	753	752	2.00003	499,628
10	19,572	19,480	19,512	19,521	897	887	892	892	2.00004	486,885
Min	122	116	119	119	14	14	14	14	2.00002	10,638
Mean	13,400	13,374	13,400	13,391	558	556	558	557	2.00003	340,324
Median	4,994	4,973	5,003	4,990	246	245	246	245	2.00003	172,838
Max	58,456	58,441	58,390	58,429	2,156	2,154	2,154	2,154	2.00005	1,162,683

Table 39.e Test Results of Incremental Effect of Enforcing Node-Degree Constraints for = 50.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	108,028	110,449	137,951	118,809	3,842	3,923	6,987	4,917	2.00108	1,508,032
02	629,754	635,799	663,146	642,900	22,121	22,275	32,607	25,667	2.00196	7,596,212
03	480,592	480,103	492,276	484,324	16,426	16,394	23,767	18,862	2.00186	5,513,749
04	6,643	6,710	8,307	7,220	418	417	663	499	2.00267	306,845
05	155,699	156,200	174,768	162,222	5,447	5,472	9,194	6,705	2.00104	1,965,157
06	138,762	137,979	177,974	151,572	4,976	4,980	9,334	6,430	2.00258	2,007,687
07	181,113	181,222	183,882	182,072	6,439	6,441	9,610	7,497	2.00144	2,273,111
08	43,367	43,234	40,384	42,328	1,589	1,600	2,404	1,864	2.00150	677,770
09	92,820	1,600	97,970	64,130	3,306	3,314	5,173	3,931	2.00218	1,280,190
10	235,560	234,574	235,669	235,268	8,175	8,201	11,947	9,441	2.00111	2,822,918
Min	6,643	1,600	8,307	7,220	418	417	663	499	2.00104	306,845
Mean	207,234	198,787	221,233	209,084	7,274	7,302	11,169	8,581	2.00174	2,595,167
Median	147,231	147,090	176,371	156,897	5,211	5,226	9,264	6,567	2.00168	1,986,422
Max	629,754	635,799	663,146	642,900	22,121	22,275	32,607	25,667	2.00267	7,596,212

Table 40. Summary of Incremental Effect of Enforcing Node-Degree Constraints

	Speedup			LP Upper Bound Improvement
	Ave. CPU Time	Ticks	Ave. Real Time	
		n = 10		
Min	0.36	0.35	0.50	0.00%
Mean	**1.41**	**1.01**	**1.16**	0.00%
Median	**1.22**	**1.04**	**1.18**	0.00%
Max	**3.21**	**1.96**	**1.85**	0.00%
		n = 20		
Min	0.64	0.52	0.66	0.00%
Mean	**1.10**	**1.01**	**1.09**	0.01%
Median	**1.15**	**0.98**	**1.05**	0.00%
Max	**1.35**	**1.44**	**1.65**	0.04%
		n = 30		
Min	0.63	0.55	0.45	0.00%
Mean	**1.64**	**1.38**	**1.35**	0.00%
Median	**1.51**	**1.29**	**1.27**	0.00%
Max	**3.24**	**2.51**	**2.34**	0.01%
		n = 40		
Min	0.91	0.72	0.82	0.00%
Mean	**1.82**	**1.55**	**1.64**	0.00%
Median	**1.44**	**1.32**	**1.38**	0.00%
Max	**4.84**	**3.80**	**4.08**	0.00%
		n = 50		
Min	0.77	0.83	0.69	0.00%
Mean	**2.50**	**1.84**	**1.76**	0.00%
Median	**1.97**	**1.79**	**1.48**	0.00%
Max	**9.28**	**3.18**	**4.76**	0.03%

Table 41. CI and GCI of Enforcing Node-Degree Constraints

n	CPU	Ticks	Real Time	CI	GCI
10	**1.27**	**1.04**	**1.18**	**1.15**	
20	**1.14**	0.98	**1.06**	**1.05**	
30	**1.54**	**1.31**	**1.29**	**1.37**	**1.43**
40	**1.54**	**1.38**	**1.45**	**1.45**	
50	**2.13**	**1.80**	**1.56**	**1.80**	

Conclusion: After applying technique 2, enforce node-degree constraints, the grand composite index of speedups (GCI) was 1.43, which means, on average, the model with enforce node-degree constraints was solved 1.43 times faster than the incumbent model. Thus, we adopted it.

5.4 Technique 3：Single-Node Demand Cuts

Tables 42, 43, and 44 summarize the results of applying the single-node demand cuts described in Section 4.5 to the incumbent triples model.

Table 42.a Incremental Effect Single-Node Demand Cuts for Triples Model n =10.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	1.60	1.91	2.69	2.07	0.31	0.36	0.40	0.36	2.00000	40.38
02	1.88	1.27	2.74	1.96	0.20	0.26	0.32	0.26	2.00000	70.39
03	2.21	0.93	1.35	1.50	0.22	0.21	0.31	0.25	2.00000	51.98
04	2.84	1.57	2.83	2.42	0.24	0.26	0.30	0.27	2.00000	49.50
05	1.68	1.59	3.57	2.28	0.30	0.26	0.33	0.30	2.00000	52.67
06	3.00	3.35	2.05	2.80	0.66	0.55	0.49	0.57	2.00000	56.49
07	1.74	1.65	2.61	2.00	0.57	0.67	0.43	0.56	2.00000	45.07
08	4.62	1.25	0.70	2.19	0.36	0.30	0.16	0.27	2.00000	91.48
09	3.84	2.80	1.85	2.83	0.60	0.59	0.57	0.59	2.00000	28.55
10	0.49	2.57	3.00	2.02	0.08	0.15	0.18	0.14	2.00000	19.43
Min	0.49	0.93	0.70	1.50	0.08	0.15	0.16	0.14	2.00000	19.43
Mean	2.39	1.89	2.34	2.21	0.35	0.36	0.35	0.35	2.00000	50.59
Median	2.04	1.62	2.65	2.13	0.31	0.28	0.33	0.29	2.00000	50.74
Max	4.62	3.35	3.57	2.83	0.66	0.67	0.57	0.59	2.00000	91.48

Table 42.b Incremental Effect Single-Node Demand Cuts for Triples Model n = 20.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	8.36	8.18	8.01	8.18	2.52	2.44	2.48	2.48	2.00024	1,017.50
02	6.04	6.60	6.62	6.42	2.92	3.08	3.23	3.08	2.00012	887.88
03	18.25	18.66	18.64	18.52	4.71	4.89	4.80	4.80	2.00060	3,190.64
04	23.01	21.58	21.54	22.04	2.94	2.82	2.85	2.87	2.00063	1,293.59
05	7.97	7.61	7.71	7.76	3.85	3.53	3.76	3.71	2.00000	1,098.06
06	40.74	40.57	39.77	40.36	5.14	5.02	4.77	4.98	2.00120	2,652.09
07	6.25	6.38	6.45	6.36	1.19	1.25	1.24	1.23	2.00024	912.88
08	7.04	7.14	6.22	6.80	2.59	2.51	2.15	2.42	2.00000	903.82
09	33.34	33.59	34.38	33.77	5.49	5.56	5.65	5.57	2.00120	3,731.75
10	15.47	15.66	15.58	15.57	4.92	5.02	4.86	4.93	2.00000	1,746.58
Min	6.04	6.38	6.22	6.36	1.19	1.25	1.24	1.23	2.00000	887.88
Mean	16.65	16.60	16.49	16.58	3.63	3.61	3.58	3.61	2.00042	1,743.48
Median	11.92	11.92	11.79	11.88	3.40	3.31	3.50	3.40	2.00024	1,195.83
Max	40.74	40.57	39.77	40.36	5.49	5.56	5.65	5.57	2.00120	3,731.75

Table 42.c Incremental Effect Single-Node Demand Cuts for Triples Model $n = 30$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	503.72	509.13	522.53	511.79	45.47	46.07	46.49	46.01	2.00000	38,712.14
02	515.72	525.54	537.24	526.16	42.70	43.32	44.02	43.35	2.00084	37,530.25
03	610.56	614.69	627.78	617.68	53.95	54.39	54.91	54.42	2.00076	50,643.76
04	64.27	65.50	65.27	65.01	11.98	12.02	12.04	12.01	2.00103	9,870.53
05	359.08	361.42	374.17	364.89	32.04	32.29	32.90	32.41	2.00104	26,453.56
06	42.00	43.85	43.89	43.25	10.01	10.94	10.89	10.61	2.00146	6,480.00
07	200.72	202.79	205.35	202.95	17.85	17.85	18.21	17.97	2.00201	15,115.31
08	247.45	249.41	250.16	249.01	27.44	27.85	27.71	27.67	2.00124	25,631.55
09	937.22	936.55	945.48	939.75	76.53	76.65	76.70	76.63	2.00151	72,046.28
10	639.46	652.14	668.38	653.33	38.27	38.76	38.71	38.58	2.00116	28,822.55
Min	42.00	43.85	43.89	43.25	10.01	10.94	10.89	10.61	2.00000	6,480.00
Mean	412.02	416.10	424.03	417.38	35.62	36.01	36.26	35.97	2.00110	31,130.59
Median	431.40	435.28	448.35	438.34	35.16	35.53	35.81	35.50	2.00110	27,638.06
Max	937.22	936.55	945.48	939.75	76.53	76.65	76.70	76.63	2.00201	72,046.28

Table 42.d Incremental Effect Single-Node Demand Cuts for Triples Model $n = 40$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	1,789	1,857	1,975	1,874	117	121	124	121	2.00002	93,185
02	39,662	41,683	43,428	41,591	1,504	1,555	1,607	1,555	2.00002	838,400
03	1,310	1,359	1,443	1,371	93	96	99	96	2.00003	78,457
04	1,184	1,206	1,227	1,206	153	153	153	153	2.00004	153,586
05	1,600	1,657	1,791	1,683	134	134	140	136	2.00005	116,150
06	120	123	123	122	16	17	17	17	2.00005	11,778
07	5,701	5,975	6,452	6,043	278	291	306	292	2.00002	190,329
08	16,448	17,265	18,197	17,303	702	723	759	728	2.00003	469,779
09	10,445	10,902	11,571	10,973	457	478	503	479	2.00003	315,020
10	17,805	18,599	19,503	18,636	750	779	806	778	2.00004	480,756
Min	120	123	123	122	16	17	17	17	2.00002	11,778
Mean	9,606	10,063	10,571	10,080	420	435	451	435	2.00003	274,744
Median	3,745	3,916	4,214	3,958	215	222	230	222	2.00003	171,958
Max	39,662	41,683	43,428	41,591	1,504	1,555	1,607	1,555	2.00005	838,400

Table 42.e Incremental Effect Single-Node Demand Cuts for Triples Model $n = 50$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	84,724	88,331	92,498	88,518	3,040	3,170	3,320	3,176	2.00108	1,295,474
02	159,434	164,355	171,481	165,090	5,568	5,735	5,969	5,757	2.00196	2,310,404
03	72,500	75,529	78,732	75,587	2,689	2,721	2,850	2,753	2.00186	1,209,763
04	8,252	8,883	9,592	8,909	467	492	508	489	2.00267	329,474
05	14,645	15,460	16,519	15,541	609	645	679	644	2.00104	316,748
06	77,032	80,702	85,150	80,961	2,745	2,837	3,006	2,863	2.00258	1,194,310
07	65,166	68,697	71,285	68,383	2,413	2,515	2,625	2,518	2.00144	1,218,643
08	6,860	7,246	7,703	7,270	347	359	378	361	2.00150	215,113
09	18,669	19,809	21,402	19,960	769	806	859	811	2.00218	400,788
10	114,244	118,440	123,559	118,748	4,079	4,198	4,412	4,229	2.00111	1,855,890
Min	6,860	7,246	7,703	7,270	347	359	378	361	2.00104	215,113
Mean	62,153	64,745	67,792	64,897	2,272	2,348	2,460	2,360	2.00174	1,034,661
Median	68,833	72,113	75,009	71,985	2,551	2,618	2,737	2,635	2.00168	1,202,037
Max	159,434	164,355	171,481	165,090	5,568	5,735	5,969	5,757	2.00267	2,310,404

Table 43. Summary of Incremental Effect Single-Node Demand Cuts for Triples Model

	Speedup			LP Upper Bound Improvement
	Ave. CPU Time	Ticks	Ave. Real Time	
$n = 10$				
Min	0.58	0.60	0.30	0.00%
Mean	**1.26**	1.00	**1.09**	0.00%
Median	**1.03**	**1.04**	**1.03**	0.00%
Max	**2.87**	**1.35**	**1.71**	0.00%
$n = 20$				
Min	0.72	0.70	0.56	0.00%
Mean	**1.20**	0.98	0.95	0.00%
Median	**1.04**	0.91	0.86	0.00%
Max	**2.53**	**1.89**	**1.94**	0.00%
$n = 30$				
Min	0.66	0.52	0.59	0.00%
Mean	**1.28**	**1.01**	**1.10**	0.00%
Median	**1.17**	0.82	0.97	0.00%
Max	**2.17**	**2.01**	**1.86**	0.00%
$n = 40$				
Min	0.71	0.54	0.56	0.00%
Mean	**1.35**	**1.16**	**1.18**	0.00%
Median	**1.16**	**1.15**	**1.20**	0.00%
Max	**2.68**	**1.59**	**1.77**	0.00%
$n = 50$				
Min	0.81	0.93	1.02	0.00%
Mean	**3.84**	**2.76**	**4.17**	0.00%
Median	**2.94**	**2.51**	**3.72**	0.00%
Max	**10.44**	**6.20**	**10.40**	0.00%

Table 44. CI and GCI of Incremental Effect Single-Node Demand Cuts for Triples Model

n	CPU	Ticks	Real Time	CI	GCI
10	**1.09**	**1.03**	**1.04**	**1.05**	
20	**1.09**	0.93	0.88	0.96	
30	**1.20**	0.86	1.00	1.00	**1.64**
40	**1.21**	**1.15**	**1.20**	**1.18**	
50	**3.17**	**2.58**	**3.85**	**3.20**	

Conclusion: After applying technique 3, single-node demand cuts, the grand composite index of speedups (GCI) was 1.64, which means, on average, the model with single-node demand cuts was solved 1.64 times faster than the incumbent model. Thus, we adopted it.

5.5 Technique 4：Branching Priority

Tables 45, 46, and 47 summarize the results of applying the branching priority described in Section 4.7 to the incumbent triples model.

Table 45.a Incremental Effect of Branching Priority for Triples Model $n = 10$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	1.16	1.73	2.51	1.80	0.23	0.38	0.39	0.33	2.00000	40.39
02	2.83	1.57	2.81	2.41	0.26	0.28	0.33	0.29	2.00000	70.39
03	2.39	3.44	2.41	2.75	0.30	0.50	0.31	0.37	2.00000	51.98
04	2.03	1.14	1.61	1.59	0.34	0.20	0.22	0.25	2.00000	49.51
05	2.76	1.65	3.61	2.67	0.30	0.29	0.34	0.31	2.00000	52.68
06	2.39	2.13	1.55	2.02	0.33	0.50	0.36	0.40	2.00000	56.50
07	3.62	1.51	4.19	3.11	0.59	0.73	0.55	0.62	2.00000	40.45
08	0.95	1.97	1.58	1.50	0.20	0.25	0.23	0.23	2.00000	91.48
09	3.63	1.83	1.89	2.45	0.53	0.57	0.33	0.48	2.00000	28.55
10	1.09	0.97	1.21	1.09	0.08	0.11	0.10	0.10	2.00000	19.43
Min	0.95	0.97	1.21	1.09	0.08	0.11	0.10	0.10	2.00000	19.43
Mean	2.28	1.79	2.34	2.14	0.32	0.38	0.32	0.34	2.00000	50.14
Median	2.39	1.69	2.15	2.22	0.30	0.34	0.33	0.32	2.00000	50.75
Max	3.63	3.44	4.19	3.11	0.59	0.73	0.55	0.62	2.00000	91.48

Table 45.b Incremental Effect of Branching Priority for Triples Model $n = 20$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	7.35	8.02	8.32	7.90	2.28	2.59	2.62	2.50	2.00024	1,029.24
02	6.94	6.64	6.53	6.70	2.55	3.34	3.42	3.10	2.00012	1,025.05
03	17.95	18.36	18.70	18.34	4.54	4.85	4.82	4.74	2.00060	2,986.63
04	33.84	15.74	16.91	22.16	3.56	2.96	3.29	3.27	2.00063	1,191.60
05	7.79	9.10	9.20	8.70	2.41	4.02	4.10	3.51	2.00000	1,134.01
06	59.24	39.31	38.65	45.73	7.16	5.21	5.32	5.90	2.00120	2,975.40
07	40.33	6.56	6.07	17.65	3.36	1.21	1.16	1.91	2.00024	912.91
08	7.50	7.38	7.76	7.55	2.08	2.69	2.98	2.58	2.00000	983.31
09	86.61	35.69	34.16	52.15	8.78	5.86	5.53	6.72	2.00120	3,901.59
10	59.73	14.57	14.96	29.75	5.90	4.34	4.68	4.97	2.00000	1,428.35
Min	6.94	6.56	6.07	6.70	2.08	1.21	1.16	1.91	2.00000	912.91
Mean	32.73	16.14	16.13	21.66	4.26	3.71	3.79	3.92	2.00042	1,756.81
Median	25.89	11.84	12.08	17.99	3.46	3.68	3.76	3.39	2.00024	1,162.81
Max	86.61	39.31	38.65	52.15	8.78	5.86	5.53	6.72	2.00120	3,901.59

Table 45.c Incremental Effect of Branching Priority for Triples Model $n = 30$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	501	428	419	449	55	45	44	48	2.00000	36,550
02	579	513	512	535	53	46	46	49	2.00084	42,555
03	912	485	481	626	89	43	43	58	2.00076	38,059
04	63	64	64	64	14	15	15	15	2.00103	12,265
05	412	333	328	358	40	33	32	35	2.00104	23,784
06	49	43	42	45	10	10	10	10	2.00146	6,348
07	507	475	472	485	47	44	43	45	2.00201	36,954
08	266	246	247	253	26	25	25	25	2.00124	22,520
09	1,223	831	836	963	96	68	70	78	2.00151	61,052
10	1,771	1,353	1,346	1,490	103	74	75	84	2.00116	60,572
Min	49	43	42	45	10	10	10	10	2.00000	6,348
Mean	629	477	475	527	53	40	40	45	2.00110	34,066
Median	504	451	445	467	50	43	43	46	2.00110	36,752
Max	1,771	1,353	1,346	1,490	103	74	75	84	2.00201	61,052

Table 45.d Incremental Effect of Branching Priority for Triples Model $n = 40$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	3,229	2,770	2,688	2,895	184	157	155	165	2.00002	112,430
02	663	450	438	517	53	32	31	39	2.00002	24,087
03	1,993	1,549	1,537	1,693	179	136	134	150	2.00003	115,872
04	2,721	1,465	1,437	1,875	297	155	153	202	2.00004	147,957
05	3,676	1,907	1,860	2,481	307	128	126	187	2.00005	99,149
06	410	144	145	233	41	17	18	25	2.00005	13,648
07	9,354	5,751	5,480	6,862	506	262	253	340	2.00002	158,469
08	14,316	9,293	8,959	10,856	697	368	356	474	2.00003	192,972
09	21,827	13,467	13,085	16,126	1,091	569	556	739	2.00003	346,382
10	23,637	14,657	14,377	17,557	1,302	623	613	846	2.00004	374,733
Min	410	144	145	233	41	17	18	25	2.00002	13,648
Mean	8,182	5,145	5,001	6,109	466	245	240	317	2.00003	158,570
Median	3,452	2,338	2,274	2,688	302	156	154	194	2.00003	131,915
Max	23,637	14,657	14,377	17,557	1,302	623	613	846	2.00005	374,733

Table 45.e Incremental Effect of Branching Priority for Triples Model for $n = 50$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	158,883	91,719	90,821	113,808	7,776	3,292	3,319	4,796	2.00108	1,493,511
02	266,752	157,822	159,416	194,663	12,546	5,548	5,543	7,879	2.00196	2,114,350
03	109,610	77,547	77,711	88,289	4,893	2,738	2,745	3,459	2.00186	1,047,296
04	6,231	3,849	3,736	4,605	646	329	324	433	2.00267	280,329
05	32,187	18,066	17,589	22,614	2,015	717	699	1,144	2.00104	327,999
06	132,681	79,786	79,954	97,474	6,108	2,805	2,778	3,897	2.00258	1,105,471
07	132,983	86,835	87,378	102,399	5,820	3,011	3,003	3,945	2.00144	1,111,524
08	16,382	7,360	7,055	10,266	1,425	433	418	759	2.00150	293,061
09	143,116	89,635	89,482	107,411	6,886	3,131	3,105	4,374	2.00218	1,361,881
10	215,784	136,825	138,530	163,713	9,414	4,949	5,025	6,463	2.00111	2,112,308
Min	6,231	3,849	3,736	4,605	646	329	324	433	2.00104	280,329
Mean	121,461	74,944	75,167	90,524	5,753	2,695	2,696	3,715	2.00174	1,124,773
Median	132,832	83,310	83,666	99,936	5,964	2,908	2,890	3,921	2.00168	1,108,497
Max	266,752	157,822	159,416	194,663	12,546	5,548	5,543	7,879	2.00267	2,114,350

Table 46. Summary of Incremental Effect of Branching Priority for Triples Model

	Speedup			LP Upper Bound Improvement
	Ave. CPU Time	Ticks	Ave. Real Time	
n = 10				
Min	0.55	1.00	0.69	0.00%
Mean	**1.14**	**1.01**	**1.11**	0.00%
Median	**1.15**	1.00	**1.14**	0.00%
Max	**1.85**	**1.11**	**1.86**	0.00%
n = 20				
Min	0.36	0.87	0.64	0.00%
Mean	0.82	1.00	0.92	0.00%
Median	0.90	0.98	0.96	0.00%
Max	**1.04**	**1.22**	**1.06**	0.00%
n = 30				
Min	0.42	0.41	0.40	0.00%
Mean	0.89	0.94	0.85	0.00%
Median	0.98	**1.04**	0.93	0.00%
Max	**1.14**	**1.33**	**1.10**	0.00%
n = 40				
Min	0.52	0.68	0.64	0.00%
Mean	0.84	**1.16**	0.83	0.00%
Median	0.68	**1.04**	0.73	0.00%
Max	**1.59**	**2.43**	**1.54**	0.00%
n = 50				
Min	0.19	0.29	0.19	0.00%
Mean	0.82	0.93	0.66	0.00%
Median	0.75	**1.02**	0.66	0.00%
Max	**1.93**	**1.18**	**1.13**	0.00%

Table 47. CI and GCI of Incremental Effect of Branching Priority for Triples Model

n	CPU	Ticks	Real Time	CI	GCI
10	**1.15**	1.00	**1.14**	**1.09**	0.91
20	0.88	0.98	0.95	0.94	
30	0.96	1.02	0.91	0.96	
40	0.72	1.07	0.76	0.86	
50	0.77	1.00	0.66	0.81	

Conclusion: After applying technique 4, branching priority, the grand composite index of speedups (GCI) was 0.91, indicating that was more efficient to solve the incumbent model. Therefore, we did not adopt this technique.

5.6 Technique 5：Lifted MTZ

Tables 48, 49, and 50 summarize the results of lifting the MTZ constraints as described in section 4.8 in the incumbent triples model.

Table 48.a Incremental Effect of Lifted MTZ in Triples Model for *n* =10.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	5.16	1.27	1.98	2.80	0.51	0.32	0.35	0.39	2.00000	32.35
02	1.78	1.47	2.36	1.87	0.27	0.32	0.36	0.32	2.00000	71.29
03	1.28	2.20	2.65	2.05	0.32	0.50	0.35	0.39	2.00000	69.81
04	1.32	1.39	2.56	1.76	0.32	0.18	0.20	0.23	2.00000	29.01
05	1.38	2.05	2.21	1.88	0.45	0.55	0.21	0.40	2.00000	40.50
06	3.49	1.76	4.20	3.15	0.44	0.49	0.51	0.48	2.00000	32.80
07	0.97	1.29	1.80	1.35	0.21	0.26	0.31	0.26	2.00000	62.57
08	1.17	1.56	2.92	1.88	0.31	0.25	0.31	0.29	2.00000	97.91
09	3.08	2.24	3.11	2.81	0.62	0.57	0.55	0.58	2.00000	35.28
10	3.16	0.91	1.27	1.78	0.18	0.11	0.12	0.14	2.00000	17.82
Min	0.97	0.91	1.27	1.35	0.18	0.11	0.12	0.14	2.00000	17.82
Mean	2.28	1.61	2.51	2.13	0.36	0.36	0.33	0.35	2.00000	48.93
Median	1.58	1.52	2.46	1.88	0.32	0.32	0.33	0.35	2.00000	37.89
Max	5.16	2.24	4.20	3.15	0.62	0.57	0.55	0.58	2.00000	97.91

Table 48.b Incremental Effect of Lifted MTZ in Triples Model for *n* =20.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	7.88	7.86	7.56	7.77	1.48	1.41	1.35	1.41	2.00024	1,153.88
02	5.45	5.57	5.55	5.52	2.53	2.71	2.56	2.60	2.00012	768.35
03	22.64	22.77	23.15	22.85	5.75	5.84	5.93	5.84	2.00060	3,184.01
04	21.46	20.31	20.55	20.77	3.25	3.24	3.25	3.25	2.00063	1,299.20
05	9.96	9.71	9.78	9.82	3.62	3.46	3.58	3.55	2.00000	1,198.43
06	47.84	47.01	46.94	47.27	5.19	4.99	4.92	5.03	2.00120	2,598.90
07	6.08	5.89	5.43	5.80	1.31	1.27	1.20	1.26	2.00024	1,015.31
08	10.45	11.43	11.33	11.07	2.22	2.31	2.38	2.30	2.00000	946.58
09	38.13	37.83	38.49	38.15	5.42	5.02	5.38	5.27	2.00120	3,228.06
10	20.65	20.32	20.64	20.54	7.70	7.59	7.81	7.70	2.00000	2,082.89
Min	5.45	5.57	5.43	5.52	1.31	1.27	1.20	1.26	2.00000	768.35
Mean	19.05	18.87	18.94	18.96	3.85	3.78	3.84	3.82	2.00042	1,747.56
Median	15.55	15.87	15.94	15.80	3.44	3.35	3.42	3.40	2.00024	1,248.82
Max	47.84	47.01	46.94	47.27	7.70	7.59	7.81	7.70	2.00120	3,228.06

Table 48.c Incremental Effect of Lifted MTZ in Triples Model for $n = 30$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	415	411	414	413	50	49	50	50	2.00000	38,411
02	510	511	517	513	42	42	43	42	2.00084	35,978
03	393	391	397	394	42	41	42	42	2.00076	36,968
04	62	61	64	63	17	17	17	17	2.00103	12,387
05	296	302	304	300	32	32	33	32	2.00104	24,524
06	43	44	43	43	12	13	12	12	2.00146	6,585
07	238	235	235	236	20	20	21	20	2.00201	18,269
08	305	307	309	307	31	31	32	31	2.00124	29,221
09	1,188	1,201	1,208	1,199	80	81	80	80	2.00151	70,949
10	1,070	1,089	1,101	1,087	64	64	65	64	2.00116	50,361
Min	43	44	43	43	12	13	12	12	2.00000	6,585
Mean	452	455	459	456	39	39	39	39	2.00111	32,365
Median	349	349	353	350	37	37	37	37	2.00110	32,600
Max	1,188	1,201	1,208	1,199	80	81	80	80	2.00201	70,949

Table 48.d Incremental Effect of Lifted MTZ in Triples Model for $n = 40$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	3,566	3,752	3,805	3,708	215	222	225	221	2.00002	171,624
02	38,795	40,334	40,529	39,886	1,469	1,531	1,528	1,510	2.00002	827,796
03	167	172	170	170	18	18	18	18	2.00003	13,932
04	403	401	406	403	36	35	36	35	2.00004	29,858
05	2,409	2,541	2,601	2,517	155	156	160	157	2.00005	121,821
06	106	110	110	109	13	13	13	13	2.00005	10,090
07	5,539	5,819	5,933	5,764	251	260	268	260	2.00002	162,603
08	16,462	17,096	17,370	16,976	672	692	709	691	2.00003	422,257
09	11,816	12,297	12,487	12,200	495	510	521	509	2.00003	314,160
10	13,900	14,297	14,535	14,244	678	689	697	688	2.00004	498,871
Min	106	110	110	109	13	13	13	13	2.00002	10,090
Mean	9,316	9,682	9,795	9,598	400	413	418	410	2.00003	257,301
Median	4,552	4,786	4,869	4,736	233	241	247	240	2.00003	167,114
Max	38,795	40,334	40,529	39,886	1,469	1,531	1,528	1,510	2.00005	827,796

Table 48.e Incremental Effect of Lifted MTZ in Triples Model for $n = 50$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	59,542	61,296	61,796	60,878	2,180	2,206	2,252	2,213	2.00108	989,870
02	151,536	153,830	151,638	152,335	5,239	5,375	5,336	5,317	2.00196	2,234,792
03	85,861	88,013	88,642	87,505	3,049	3,121	3,132	3,101	2.00186	1,384,630
04	3,025	3,104	3,186	3,105	295	293	298	295	2.00267	258,223
05	59,747	62,436	62,635	61,606	2,317	2,387	2,421	2,375	2.00104	1,258,846
06	73,880	76,684	77,022	75,862	2,638	2,711	2,737	2,695	2.00248	1,177,145
07	72,171	74,723	75,077	73,990	2,688	2,769	2,801	2,753	2.00144	1,298,540
08	5,454	5,634	5,766	5,618	342	352	356	350	2.00150	249,632
09	75,334	77,390	77,516	76,747	2,818	2,882	2,874	2,858	2.00218	1,603,195
10	136,861	139,016	143,006	139,628	4,865	4,965	5,090	4,973	2.00111	2,258,454
Min	3,025	3,104	3,186	3,105	295	293	298	295	2.00104	249,632
Mean	72,341	74,213	74,628	73,727	2,643	2,706	2,730	2,693	2.00173	1,271,333
Median	73,026	75,703	76,050	74,926	2,663	2,740	2,769	2,724	2.00168	1,278,693
Max	151,536	153,830	151,638	152,335	5,239	5,375	5,336	5,317	2.00267	2,258,454

Table 49. Summary of Incremental Effect of Lifted MTZ Constraints on Triples Model

	Speedup			LP Upper Bound Improvement
	Ave. CPU Time	Ticks	Ave. Real Time	
$n = 10$				
Min	0.73	0.72	0.55	0.00%
Mean	**1.08**	**1.13**	**1.05**	0.00%
Median	**1.09**	**1.04**	**1.01**	0.00%
Max	**1.48**	**1.72**	**1.65**	0.00%
$n = 20$				
Min	0.61	0.84	0.64	0.00%
Mean	0.91	0.98	**1.04**	0.00%
Median	0.87	0.98	**1.02**	0.00%
Max	**1.16**	**1.16**	**1.75**	0.00%
$n = 30$				
Min	0.60	0.57	0.60	0.00%
Mean	**1.01**	0.96	0.91	0.00%
Median	**1.01**	1.00	0.90	0.00%
Max	**1.57**	**1.37**	**1.31**	0.00%
$n = 40$				
Min	0.51	0.54	0.55	0.00%
Mean	**1.87**	**1.87**	**1.75**	0.00%
Median	**1.05**	**1.06**	**1.09**	0.00%
Max	**8.08**	**5.63**	**5.28**	0.00%
$n = 50$				
Min	0.25	0.25	0.27	0.00%
Mean	**1.09**	0.86	0.95	0.00%
Median	1.00	0.91	0.97	0.00%
Max	**2.87**	**1.31**	**1.65**	0.00%

Table 50. CI and GCI of Lifted MTZ in Triples Model

n	CPU	Ticks	Real Time	CI	GCI
10	**1.09**	**1.06**	**1.02**	**1.05**	
20	0.88	0.98	1.02	0.97	
30	**1.02**	0.99	0.91	0.97	**1.04**
40	**1.27**	**1.26**	**1.25**	**1.26**	
50	**1.03**	0.90	0.97	0.96	

Conclusion: After applying technique 5, lifted MTZ, the grand composite index of speedups (GCI) is 1.04, which means, on average, the model with lifted MTZ was 1.04 times faster than the incumbent model. Thus, we adopted it.

5.7 Technique 6：MTZ upper bound

Tables 51, 52, and 53 summarize the effects of imposing a bound on the MTZ sequence variables as described in Section 4.9 on the triples model.

Table 51.a Incremental Effect of MTZ upper bound in Triples Model for *n* =10.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	1.79	2.31	1.06	1.72	0.30	0.32	0.29	0.30	2.00000	34.71
02	1.53	2.48	1.56	1.85	0.21	0.25	0.29	0.25	2.00000	65.64
03	3.32	3.93	1.74	3.00	0.49	0.49	0.48	0.49	2.00000	54.90
04	2.15	2.31	1.83	2.10	0.35	0.36	0.31	0.34	2.00000	42.70
05	2.03	2.79	1.18	2.00	0.20	0.23	0.15	0.19	2.00000	41.03
06	3.56	2.94	1.88	2.79	1.02	1.15	1.01	1.06	2.00000	38.94
07	1.62	1.71	2.55	1.96	0.42	0.54	0.53	0.50	2.00000	41.14
08	1.56	2.50	1.42	1.83	0.21	0.24	0.18	0.21	2.00000	84.09
09	1.23	0.66	1.61	1.16	0.16	0.11	0.19	0.15	2.00000	29.73
10	1.06	1.16	0.44	0.89	0.10	0.09	0.07	0.09	2.00000	21.80
Min	1.06	0.66	0.44	0.89	0.10	0.09	0.07	0.09	2.00000	21.80
Mean	1.98	2.28	1.53	1.93	0.35	0.38	0.35	0.36	2.00000	45.47
Median	1.71	2.39	1.58	1.91	0.26	0.29	0.29	0.28	2.00000	41.09
Max	3.56	3.93	2.55	3.00	1.02	1.15	1.01	1.06	2.00000	84.09

Table 51.b Incremental Effect of MTZ upper bound in Triples Model for n =20.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	12.25	13.10	10.24	11.87	3.90	4.01	3.62	3.84	2.00024	1,864.56
02	6.60	9.28	5.56	7.15	2.25	2.30	2.37	2.31	2.00012	757.58
03	10.70	10.99	11.33	11.00	2.86	2.90	2.88	2.88	2.00060	1,298.64
04	17.76	18.00	17.43	17.73	3.01	3.17	3.31	3.16	2.00063	1,335.02
05	7.92	10.42	9.52	9.29	3.41	3.32	3.83	3.52	2.00000	1,355.41
06	51.04	50.22	51.01	50.76	3.90	3.93	3.79	3.87	2.00120	2,239.13
07	7.03	7.15	7.01	7.06	1.34	1.35	1.36	1.35	2.00024	1,002.93
08	9.47	10.87	10.20	10.18	2.64	2.71	2.64	2.66	2.00000	1,079.31
09	41.62	43.28	42.21	42.37	5.97	5.76	5.95	5.89	2.00120	3,614.30
10	11.82	12.22	13.31	12.45	2.55	2.61	2.77	2.64	2.00000	1,186.95
Min	6.60	7.15	5.56	7.06	1.34	1.35	1.36	1.35	2.00000	757.58
Mean	17.62	18.55	17.78	17.99	3.18	3.21	3.25	3.21	2.00042	1,573.38
Median	11.26	11.60	10.79	11.43	2.94	3.04	3.10	3.02	2.00024	1,316.83
Max	51.04	50.22	51.01	50.76	5.97	5.76	5.95	5.89	2.00120	3,614.30

Table 51.c Incremental Effect of MTZ upper bound in Triples Model for n = 30.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	386	388	403	392	35	35	35	35	2.00000	26,475
02	555	570	570	565	52	53	52	52	2.00084	43,281
03	435	432	433	433	45	45	44	45	2.00076	38,164
04	82	80	79	80	15	15	15	15	2.00103	12,938
05	340	346	348	345	33	34	33	33	2.00104	24,637
06	42	43	42	42	10	10	10	10	2.00146	6,263
07	208	210	212	210	19	20	20	20	2.00201	17,145
08	598	604	637	613	38	39	40	39	2.00124	29,961
09	932	967	983	961	65	68	68	67	2.00151	55,735
10	1,124	1,160	1,190	1,158	72	74	75	73	2.00116	58,062
Min	42	43	42	42	10	10	10	10	2.00000	6,263
Mean	470	480	490	480	38	39	39	39	2.00110	31,266
Median	410	410	418	413	36	37	37	37	2.00110	28,218
Max	1,124	1,160	1,190	1,158	72	74	75	73	2.00201	58,062

Table 51.d Incremental Effect of MTZ upper bound in Triples Model for $n = 40$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	3,473	3,642	3,753	3,623	216	226	228	223	2.00002	166,015
02	48,563	50,988	52,396	50,649	1,805	1,865	1,918	1,863	2.00002	977,826
03	1,706	1,783	1,778	1,756	154	159	158	157	2.00003	136,353
04	334	342	342	339	83	86	86	85	2.00004	52,850
05	2,350	2,460	2,526	2,445	171	177	179	176	2.00005	140,095
06	76	76	74	75	13	14	13	13	2.00005	9,808
07	6,458	6,701	6,945	6,701	298	307	316	307	2.00002	193,303
08	10,798	11,093	11,577	11,156	433	438	460	444	2.00003	245,850
09	7,285	7,506	7,746	7,512	317	325	334	325	2.00003	189,765
10	13,229	13,844	14,225	13,766	626	649	662	645	2.00004	462,913
Min	76	76	74	75	13	14	13	13	2.00002	9,808
Mean	9,427	9,843	10,136	9,802	412	424	436	424	2.00003	257,478
Median	4,966	5,172	5,349	5,162	257	266	272	265	2.00003	177,890
Max	48,563	50,988	52,396	50,649	1,805	1,865	1,918	1,863	2.00005	977,826

Table 51.e Incremental Effect of MTZ upper bound in Triples Model for $n = 50$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	57,085	60,619	62,352	60,019	2,146	2,263	2,280	2,230	2.00108	1,154,436
02	136,413	146,583	148,175	143,724	4,767	5,104	5,178	5,016	2.00196	1,988,723
03	74,924	80,044	80,395	78,454	2,781	2,899	2,934	2,871	2.00186	1,304,192
04	3,098	3,216	3,242	3,185	356	363	360	359	2.00267	338,100
05	19,397	20,037	20,602	20,012	801	825	860	829	2.00104	404,223
06	15,499	16,313	16,550	16,121	760	792	818	790	2.00248	467,217
07	67,595	71,849	73,539	70,994	2,601	2,751	2,795	2,716	2.00144	1,398,034
08	8,482	8,857	9,227	8,855	438	455	464	453	2.00150	274,003
09	66,965	67,367	68,774	67,702	2,432	2,437	2,497	2,455	2.00218	1,214,504
10	98,926	95,167	96,668	96,920	4,007	3,930	3,970	3,969	2.00111	2,270,514
Min	3,098	3,216	3,242	3,185	356	363	360	359	2.00104	274,003
Mean	54,838	57,005	57,952	56,599	2,109	2,182	2,216	2,169	2.00173	1,081,395
Median	62,025	63,993	65,563	63,860	2,289	2,350	2,388	2,342	2.00168	1,184,470
Max	136,413	146,583	148,175	143,724	4,767	5,104	5,178	5,016	2.00267	2,270,514

Table 52. Summary of Incremental Effect of **MTZ upper bound** Constraints

	Speedup			LP Upper Bound Improvement
	Ave. CPU Time	Ticks	Ave. Real Time	
n = 10				
Min	0.68	0.68	0.45	0.00%
Mean	**1.24**	**1.05**	**1.39**	0.00%
Median	**1.02**	**1.04**	**1.28**	0.00%
Max	**2.41**	**1.52**	**3.78**	0.00%
n = 20				
Min	0.65	0.62	0.37	0.00%
Mean	**1.11**	**1.16**	**1.25**	0.00%
Median	0.99	0.99	**1.02**	0.00%
Max	**2.08**	**2.45**	**2.91**	0.00%
n = 30				
Min	0.50	0.83	0.81	0.00%
Mean	0.93	**1.04**	**1.04**	0.00%
Median	0.92	0.99	1.00	0.00%
Max	**1.25**	**1.45**	**1.44**	0.01%
n = 40				
Min	0.10	0.10	0.12	0.00%
Mean	**1.06**	0.97	0.93	0.00%
Median	**1.03**	0.95	0.94	0.00%
Max	**1.62**	**1.72**	**1.56**	0.00%
n = 50				
Min	0.63	0.76	0.77	0.01%
Mean	**1.62**	**1.36**	**1.44**	0.01%
Median	**1.09**	**1.03**	**1.07**	0.01%
Max	**4.71**	**3.11**	**3.41**	0.03%

Table 53. CI and GCI of MTZ Upper Bound in Triples Model

n	CPU	Ticks	Real Time	CI	GCI
10	**1.07**	**1.04**	**1.32**	**1.15**	
20	**1.02**	**1.04**	**1.07**	**1.05**	
30	0.92	1.00	**1.01**	0.98	**1.06**
40	**1.03**	0.95	0.94	0.97	
50	**1.22**	**1.11**	**1.16**	**1.16**	

Conclusion: After applying technique 6, MTZ upper bound, the grand composite index of speedups (GCI) was 1.06, which means, on average, the model with MTZ upper bound was 1.06 times faster than the incumbent model. Thus, we adopted it.

5.8 Technique 7：Pairwise Demand Cuts

Tables 54, 55, and 56 summarize the effects of the pairwise demand cuts described in Section 4.11 on the incumbent triples model.

Table 54.a Incremental Effect of Pairwise Demand Cuts in Triples Model for $n = 10$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	1.64	1.95	1.94	1.84	0.49	0.55	0.61	0.55	2.00000	43.04
02	1.66	1.26	1.00	1.31	0.37	0.34	0.34	0.35	2.00000	72.64
03	2.35	2.30	5.73	3.46	0.38	0.52	0.48	0.46	2.00000	58.14
04	2.42	1.35	1.16	1.64	0.29	0.35	0.28	0.30	2.00000	39.17
05	2.27	1.45	1.05	1.59	0.29	0.36	0.33	0.33	2.00000	43.40
06	3.56	1.55	1.82	2.31	0.48	0.47	0.44	0.46	2.00000	44.02
07	1.65	1.91	3.66	2.41	0.30	0.50	0.44	0.41	2.00000	77.99
08	2.61	1.36	1.19	1.72	0.32	0.43	0.38	0.38	2.00000	67.88
09	1.99	1.20	1.79	1.66	0.39	0.35	0.38	0.38	2.00000	32.95
10	0.85	0.95	1.20	1.00	0.17	0.18	0.21	0.19	2.00000	21.30
Min	0.85	0.95	1.00	1.00	0.17	0.18	0.21	0.19	2.00000	21.30
Mean	2.10	1.53	2.05	1.89	0.35	0.41	0.39	0.38	2.00000	50.05
Median	2.13	1.41	1.50	1.69	0.35	0.39	0.38	0.38	2.00000	43.71
Max	3.56	2.30	5.73	3.46	0.49	0.55	0.61	0.55	2.00000	77.99

Table 54.b Incremental Effect of Pairwise Demand Cuts in Triples Model for $n = 20$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	11.93	10.84	10.72	11.16	2.62	2.36	2.36	2.44	2.00024	1,195.60
02	8.61	8.70	8.56	8.62	3.09	3.42	3.31	3.28	2.00012	871.91
03	24.87	24.59	24.25	24.57	6.05	6.24	6.10	6.13	2.00060	3,359.75
04	24.18	23.93	23.97	24.03	3.04	3.10	2.98	3.04	2.00063	1,276.76
05	11.44	12.13	10.72	11.43	4.02	4.57	4.30	4.29	2.00000	1,326.97
06	52.51	50.90	53.63	52.34	4.94	5.09	4.97	5.00	2.00120	2,393.68
07	9.30	9.08	9.57	9.32	1.89	1.86	1.77	1.84	2.00024	1,069.51
08	10.43	11.38	10.54	10.78	2.74	2.99	2.87	2.87	2.00000	971.54
09	44.77	43.63	44.01	44.13	7.84	7.56	7.54	7.65	2.00120	4,668.66
10	13.50	13.22	13.46	13.39	3.17	3.09	3.15	3.14	2.00000	1,231.83
Min	8.61	8.70	8.56	8.62	1.89	1.86	1.77	1.84	2.00000	871.91
Mean	21.15	20.84	20.94	20.98	3.94	4.03	3.93	3.97	2.00042	1,836.62
Median	12.71	12.68	12.09	12.41	3.13	3.26	3.23	3.21	2.00024	1,254.30
Max	52.51	50.90	53.63	52.34	7.84	7.56	7.54	7.65	2.00120	4,668.66

Table 54.c Incremental Effect of Pairwise Demand Cuts in Triples Model for $n = 30$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	407	396	397	400	36	36	36	36	2.00000	24,127
02	550	537	542	543	48	47	47	47	2.00084	38,595
03	331	324	324	326	30	30	29	29	2.00076	21,074
04	80	83	82	82	17	19	18	18	2.00102	9,623
05	330	316	324	324	32	31	31	31	2.00103	22,346
06	67	66	66	66	14	14	14	14	2.00143	7,710
07	579	569	568	572	58	57	57	58	2.00195	51,896
08	308	306	306	306	28	28	28	28	2.00124	21,740
09	866	852	855	857	62	61	62	62	2.00144	50,339
10	1,271	1,236	1,250	1,252	78	76	76	76	2.00115	61,032
Min	67	66	66	66	14	14	14	14	2.00000	7,710
Mean	479	468	471	473	40	40	40	40	2.00109	30,848
Median	369	360	360	363	34	33	33	34	2.00109	23,236
Max	1,271	1,236	1,250	1,252	78	76	76	76	2.00195	61,032

Table 54.d Incremental Effect of Pairwise Demand Cuts in Triples Model for $n = 40$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	4,771	4,686	4,754	4,737	254	248	256	253	2.00002	174,779
02	47,036	47,029	48,382	47,482	1,797	1,791	1,833	1,807	2.00002	1,009,754
03	1,227	1,190	1,220	1,212	107	105	106	106	2.00003	83,875
04	390	386	382	386	40	40	39	40	2.00004	26,484
05	3,539	3,458	3,547	3,515	220	214	216	217	2.00005	160,177
06	164	156	159	160	21	20	21	21	2.00005	10,982
07	13,218	13,023	13,337	13,192	621	604	613	613	2.00002	420,735
08	16,042	15,841	16,370	16,084	704	690	712	702	2.00003	461,088
09	7,730	7,603	7,825	7,719	386	379	388	385	2.00003	256,009
10	16,209	16,074	16,566	16,283	650	638	657	648	2.00004	381,025
Min	164	156	159	160	21	20	21	21	2.00002	10,982
Mean	11,033	10,944	11,254	11,077	480	473	484	479	2.00003	298,491
Median	6,251	6,144	6,290	6,228	320	313	322	319	2.00003	215,394
Max	47,036	47,029	48,382	47,482	1,797	1,791	1,833	1,807	2.00005	1,009,754

Table 54.e Incremental Effect of Pairwise Demand Cuts in Triples Model for $n = 50$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	54,936	55,220	57,277	55,811	2,078	2,127	2,162	2,122	2.00108	1,012,679
02	171,945	177,419	179,265	176,210	6,019	6,169	6,288	6,159	2.00196	2,517,416
03	75,874	77,582	81,860	78,439	2,807	2,850	2,994	2,884	2.00186	1,356,211
04	5,226	5,180	5,420	5,275	403	390	402	398	2.00262	294,987
05	54,709	54,706	58,117	55,844	2,126	2,105	2,257	2,163	2.00098	1,044,172
06	16,469	16,458	17,068	16,665	783	779	794	785	2.00247	443,939
07	78,189	79,629	82,554	80,124	2,944	2,974	3,076	2,998	2.00143	1,418,407
08	10,651	10,780	11,130	10,854	525	526	537	529	2.00149	297,638
09	69,195	70,762	72,988	70,982	2,527	2,565	2,646	2,579	2.00214	1,222,352
10	28,431	28,755	29,753	28,979	1,213	1,214	1,239	1,222	2.00110	624,504
Min	5,226	5,180	5,420	5,275	403	390	402	398	2.00098	294,987
Mean	56,562	57,649	59,543	57,918	2,142	2,170	2,239	2,184	2.00171	1,023,231
Median	54,822	54,963	57,697	55,827	2,102	2,116	2,210	2,142	2.00167	1,028,426
Max	171,945	177,419	179,265	176,210	6,019	6,169	6,288	6,159	2.00262	2,517,416

Table 55. Summary of Incremental Effect of Pairwise Demand Cuts in Triples Model

	Speedup			LP Upper Bound Improvement
	Ave. CPU Time	Ticks	Ave. Real Time	
$n = 10$				
Min	0.70	0.53	0.41	0.00%
Mean	**1.04**	0.93	0.90	0.00%
Median	1.00	0.92	0.65	0.00%
Max	**1.42**	**1.24**	**2.29**	0.00%
$n = 20$				
Min	0.45	0.39	0.47	0.00%
Mean	0.85	0.96	0.87	0.00%
Median	0.88	0.95	0.80	0.00%
Max	**1.06**	**1.56**	**1.57**	0.00%
$n = 30$				
Min	0.37	0.33	0.34	0.00%
Mean	**1.05**	**1.11**	1.00	0.00%
Median	**1.01**	**1.10**	**1.02**	0.00%
Max	**2.00**	**1.81**	**1.52**	0.00%
$n = 40$				
Min	0.47	0.46	0.50	0.00%
Mean	0.83	**1.03**	1.00	0.00%
Median	0.81	0.92	0.86	0.00%
Max	**1.45**	**2.00**	**2.15**	0.00%
$n = 50$				
Min	0.36	0.39	0.38	0.00%
Mean	**1.08**	**1.20**	**1.11**	0.00%
Median	0.92	0.99	0.93	0.00%
Max	**3.34**	**3.64**	**3.25**	0.00%

Table 56. CI and GCI of Pairwise Demand Cuts Constraints

n	CPU	Ticks	Real Time	CI	GCI
10	**1.01**	0.92	0.72	0.87	0.95
20	0.87	0.95	0.82	0.88	
30	**1.02**	**1.10**	**1.01**	**1.05**	
40	0.81	0.95	0.90	0.89	
50	0.97	**1.05**	0.98	1.00	

Conclusion: After applying technique 7, pairwise demand cuts, the grand composite index of speedups (GCI) was 0.95, indicating that was more efficient to solve the incumbent model. Therefore, we did not adopt this technique.

5.9 Technique 8：Cover Cuts

Tables 57, 58, and 59 summarize the effects of including the cover cuts described in Section 4.10 in the incumbent triples model.

Table 57.a Incremental Effect of Cover Cuts in Triples Model for *n* =10.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	2.73	1.65	1.72	2.03	0.52	0.55	0.58	0.55	2.00000	38.95
02	3.34	1.98	2.93	2.75	0.35	0.37	0.42	0.38	2.00000	69.22
03	3.75	2.98	1.26	2.66	0.38	0.44	0.33	0.38	2.00000	58.14
04	1.93	1.65	3.29	2.29	0.27	0.33	0.32	0.30	2.00000	47.03
05	3.15	2.63	3.44	3.07	0.30	0.39	0.42	0.37	2.00000	58.05
06	3.26	2.62	2.47	2.78	1.17	1.05	1.27	1.16	2.00000	39.63
07	3.44	4.35	3.42	3.74	0.64	0.67	0.82	0.71	2.00000	42.73
08	2.99	1.79	2.82	2.53	0.46	0.37	0.53	0.46	2.00000	91.32
09	3.55	1.94	2.14	2.54	0.39	0.42	0.43	0.41	2.00000	32.97
10	2.88	2.24	1.31	2.14	0.22	0.25	0.18	0.22	2.00000	25.75
Min	1.93	1.65	1.26	2.03	0.22	0.25	0.18	0.22	2.00000	25.75
Mean	3.10	2.38	2.48	2.65	0.47	0.48	0.53	0.49	2.00000	50.38
Median	3.20	2.11	2.65	2.60	0.39	0.41	0.43	0.40	2.00000	44.88
Max	3.75	4.35	3.44	3.74	1.17	1.05	1.27	1.16	2.00000	91.32

Table 57.b Incremental Effect of Cover Cuts in Triples Model for $n = 20$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	10.08	10.08	9.80	9.99	1.74	1.50	1.69	1.64	2.00024	1,091.96
02	9.50	8.15	7.16	8.27	2.84	3.15	2.64	2.88	2.00012	830.11
03	14.87	14.06	13.24	14.06	3.60	3.66	3.51	3.59	2.00060	1,281.55
04	24.68	24.28	24.26	24.41	3.16	3.15	3.14	3.15	2.00063	1,278.14
05	12.70	11.18	11.19	11.69	3.61	3.65	3.61	3.62	2.00000	1,236.54
06	50.72	50.33	49.39	50.15	5.05	5.18	5.04	5.09	2.00120	2,633.05
07	9.15	9.48	9.33	9.32	1.59	1.53	1.45	1.53	2.00024	848.01
08	16.11	13.72	13.88	14.57	3.47	3.44	3.36	3.42	2.00000	1,124.51
09	44.87	45.46	43.65	44.66	6.34	6.51	6.21	6.35	2.00120	3,640.30
10	16.32	16.87	16.31	16.50	3.27	3.53	3.47	3.42	2.00000	1,379.11
Min	9.15	8.15	7.16	8.27	1.59	1.50	1.45	1.53	2.00000	830.11
Mean	20.90	20.36	19.82	20.36	3.47	3.53	3.41	3.47	2.00042	1,534.33
Median	15.49	13.89	13.56	14.31	3.37	3.49	3.41	3.42	2.00024	1,257.34
Max	50.72	50.33	49.39	50.15	6.34	6.51	6.21	6.35	2.00120	3,640.30

Table 57.c Incremental Effect of Cover Cuts in Triples Model for $n = 30$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	390	368	374	377	32	31	32	32	2.00000	22,285
02	551	540	530	540	46	45	44	45	2.00084	35,298
03	325	325	319	323	31	31	31	31	2.00076	26,605
04	82	82	82	82	14	14	14	14	2.00103	9,625
05	356	339	342	346	32	31	31	31	2.00104	22,640
06	63	62	60	61	12	12	12	12	2.00146	6,969
07	345	332	334	337	24	23	23	23	2.00201	17,597
08	517	504	506	509	37	37	36	37	2.00124	27,866
09	1,044	1,022	1,029	1,032	71	70	70	70	2.00151	58,433
10	1,274	1,258	1,255	1,262	73	72	72	72	2.00116	57,087
Min	63	62	60	61	12	12	12	12	2.00000	6,969
Mean	495	483	483	487	37	37	36	37	2.00110	28,440
Median	373	353	358	362	32	31	31	31	2.00110	24,623
Max	1,274	1,258	1,255	1,262	73	72	72	72	2.00201	58,433

Table 57.d Incremental Effect of Cover Cuts in Triples Model for $n = 40$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	4,300	4,188	4,276	4,255	230	226	229	228	2.00002	159,697
02	40,902	40,761	41,903	41,189	1,537	1,518	1,557	1,537	2.00002	825,500
03	1,465	1,443	1,477	1,462	138	135	137	137	2.00003	116,597
04	410	397	402	403	84	87	87	86	2.00004	54,108
05	1,926	1,893	1,924	1,914	124	120	122	122	2.00005	90,208
06	147	146	145	146	21	21	21	21	2.00005	10,434
07	5,745	5,603	5,772	5,707	267	263	270	267	2.00002	169,254
08	8,963	8,876	9,151	8,997	375	370	380	375	2.00003	217,200
09	8,859	8,639	9,046	8,848	365	362	370	365	2.00003	209,020
10	12,840	12,663	13,009	12,837	594	583	598	592	2.00004	403,339
Min	147	146	145	146	21	21	21	21	2.00002	10,434
Mean	8,556	8,461	8,710	8,576	374	368	377	373	2.00003	225,536
Median	5,022	4,896	5,024	4,981	249	244	249	247	2.00003	164,476
Max	40,902	40,761	41,903	41,189	1,537	1,518	1,557	1,537	2.00005	825,500

Table 57.e Incremental Effect of Cover Cuts in Triples Model for $n = 50$.

Instance	CPU Time Run 1	CPU Time Run 2	CPU Time Run 3	Ave. CPU Time	Real Time Run 1	Real Time Run 2	Real Time Run 3	Ave. Real Time	LP Upper Bound	Ticks
01	68,297	69,609	71,206	69,704	2,867	2,913	2,944	2,908	2.00108	1,820,955
02	138,769	141,921	146,276	142,322	4,924	5,011	5,129	5,022	2.00196	2,229,416
03	75,719	77,300	80,848	77,956	2,682	2,746	2,814	2,747	2.00186	1,142,579
04	5,809	5,806	6,066	5,894	411	404	415	410	2.00267	302,474
05	64,022	64,760	67,875	65,552	2,348	2,346	2,478	2,391	2.00104	1,205,026
06	14,488	14,613	15,450	14,850	730	733	761	741	2.00248	438,844
07	71,361	72,741	76,292	73,465	2,727	2,742	2,870	2,779	2.00144	1,361,864
08	8,312	8,382	8,802	8,499	457	454	467	459	2.00150	277,007
09	67,345	68,589	72,547	69,494	2,453	2,472	2,610	2,512	2.00218	1,146,126
10	98,784	101,344	105,507	101,878	3,808	3,819	3,991	3,873	2.00111	2,014,816
Min	5,809	5,806	6,066	5,894	411	404	415	410	2.00104	277,007
Mean	61,290	62,507	65,087	62,961	2,341	2,364	2,448	2,384	2.00173	1,193,911
Median	67,821	69,099	71,876	69,599	2,567	2,607	2,712	2,629	2.00168	1,175,576
Max	138,769	141,921	146,276	142,322	4,924	5,011	5,129	5,022	2.00267	2,229,416

Table 58. Summary of Incremental Effect of Cover Cuts in the Triples Model

	Speedup			LP Upper Bound Improvement
	Ave. CPU Time	Ticks	Ave. Real Time	
n = 10				
Min	0.41	0.71	0.37	0.00%
Mean	0.73	0.90	0.70	0.00%
Median	0.70	0.91	0.61	0.00%
Max	**1.12**	0.98	1.27	0.00%
n = 20				
Min	0.70	0.85	0.76	0.00%
Mean	0.85	**1.06**	1.00	0.00%
Median	0.79	1.00	0.84	0.00%
Max	**1.19**	**1.71**	**2.34**	0.00%
n = 30				
Min	0.62	0.90	0.83	0.00%
Mean	0.98	**1.12**	**1.05**	0.00%
Median	0.99	**1.08**	**1.05**	0.00%
Max	**1.34**	**1.43**	**1.45**	0.00%
n = 40				
Min	0.52	0.91	0.65	0.00%
Mean	1.00	**1.11**	**1.06**	0.00%
Median	**1.07**	**1.13**	**1.09**	0.00%
Max	**1.28**	**1.55**	**1.44**	0.00%
n = 50				
Min	0.31	0.34	0.35	0.00%
Mean	0.87	0.94	0.91	0.00%
Median	0.97	**1.04**	0.98	0.00%
Max	**1.09**	**1.14**	**1.07**	0.00%

Table 59. CI and GCI of C**over cut** Constraints

n	CPU	Ticks	Real Time	CI	GCI
10	0.71	0.91	0.63	0.75	0.99
20	0.80	1.02	0.89	0.91	
30	0.98	**1.09**	**1.06**	**1.05**	
40	**1.06**	**1.13**	**1.08**	**1.09**	
50	0.95	**1.02**	0.96	0.98	

Conclusion: After applying technique 8, cover cut, the grand composite index of speedups (GCI) was 0.99, indicating that was more efficient to solve the incumbent model. Therefore, we did not adopt this technique.

5.10 Summary of Enhanced Triples Model

We conclude this section by restating the enhanced triples model:

Maximize $p\left[\sum_{(k,\,l)\in A} d_{kl} w_{kl} y_{kl}\right] - c\sum_{(i,\,j)\in \mathrm{A}} \theta_{ij} d_{ij} - cv\sum_{(i,\,j)\in A} d_{ij} x_{ij}$

$\sum_{(1,j)\in A} x_{1,j} = 1$

$\sum_{(i,n)\in A} x_{i,n} = 1$

$\sum_{i\in V\setminus\{k,\,n\}} x_{ik} = \sum_{j\in V\setminus\{1,\,k\}} x_{kj}\,, \quad k \in V\setminus\{1,n\}$

$\sum_{i\in V\setminus\{k,\,n\}} x_{ik} \le 1 \;\; k \in V\setminus\{1,n\}$ (Node Degree Cuts)

$\sum_{(i,\,j)\in A} d_{ij} x_{ij} \le D$

$s_i - s_j + (n-1)\,x_{ij} + (n-3)\,x_{ji} \le n-2 \qquad \{(i,j) \in A:\, i \ne 1, j \ne \mathrm{n}\}$ (Lifted MTZ)

$\theta_{ij} = w_{ij} y_{ij} + \sum_{(i,\,k,\,j)\in T} u_{ik}^{j} + \sum_{(k,\,j,\,i)\in T} u_{kj}^{i} - \sum_{(i,\,j,\,k)\in T} u_{ij}^{k},\ (i,\,j) \in A$

$\theta_{ij} \le Q x_{ij},\ (i,\,j) \in A$

$u_{ij}^{k} \ge 0,\ (i,\,j,\,k) \in T$

$\sum_{j\in V\setminus\{1,i\}} w_{ij} y_{ij} \le Q \qquad \forall i \in V \setminus \{n\}$ (Single-Node Demand Cuts)

$\sum_{i\in V\setminus\{j,n\}} w_{ij} y_{ij} \le Q \qquad \forall j \in V \setminus \{1\}$ (Single-Node Demand Cuts)

$x_{ij} \in \{0,\,1\} \quad (i,\,j) \in A$

$y_{kl} \in \{0,\,1\} \quad (k,\,l) \in A$

$1 \le S_i \le n-1 \quad \forall i \in V \setminus \{1\}$ (MTZ upper bound)

6. Best Node-Arc vs. Best Triples Comparison

In this section we compare the enhanced node-arc and triples model on the 10-, 20-, 30-, and 40-node problem instances. Recall that the CPLEX solution statistics for the enhanced node-arc model on the 10- 20-, and 30-node, problem instances are given in Tables 19a, 19b, and 19c, respectively, and the statistics for the node-arc model on the 40-node problem instances are given in Tables 34a, 34b, and 34c. Recall also that the CPLEX solution statistics for the enhanced triples mode are given in Tables 51a-51e. The results are summarized in Table 60. In almost every case, the enhanced triples formulation was solved faster than the enhanced node-arc formulation regardless of which performance measure is considered. Furthermore, we note that the speedups for ticks, CPU and real time all show an increasing trend as problem size increases. We note a similar trend in the LP upper bound improvement.

Table 60. Best Node-Arc vs. Best Triples

Best Triples vs Best Node-arc					
n		CPU speedup	ticks speedup	real time speedup	LP upper bound improvement
10	Min	0.39	**1.47**	0.86	0.00%
	Mean	**2.38**	**4.47**	**2.31**	47.71%
	Median	**2.40**	**4.08**	**2.22**	54.52%
	Max	**4.50**	**8.38**	**4.00**	79.68%
20	Min	**6.34**	**8.89**	**7.98**	86.00%
	Mean	**62.43**	**24.02**	**21.46**	89.30%
	Median	**68.97**	**25.39**	**20.01**	89.22%
	Max	**137.06**	**42.75**	**46.20**	91.82%
30	Min	**34.33**	**16.06**	**30.96**	91.20%
	Mean	**210.55**	**52.26**	**101.59**	93.33%
	Median	**153.62**	**44.40**	**106.01**	93.81%
	Max	**872.08**	**137.89**	**212.53**	95.56%
40	Min	**44.46**	**34.79**	**56.61**	94.53%
	Mean	**1,943.68**	**319.00**	**1,050.75**	95.92%
	Median	**327.41**	**96.47**	**225.86**	96.21%
	Max	**11,929.94**	**1,417.70**	**4,275.04**	96.97%

7. CONCLUSIONS

Enhanced Node-Arc Formulation: The best techniques for node-arc formulation are: conditional arc-flow, relax node-degree, relax *x*-*z* linking, branching priority. With the best techniques, we were able to solve problem instances that were previously unsolved with the original node-arc formulation in the literature: all of the 30- and 40- node instances.

Enhanced Triples Formulation: The best techniques for triples formulation are: relax *u*-*x* (3d) linking constraints, add node-degree, single-node demand cuts, lifted MTZ, MTZ upper bound. For the triples model in the literature, the maximum tried problem size is 40 nodes. Using the original model, we solved instances with 50 nodes and the mean real time was more than 5 hours. After adding our most effective techniques, the triples model can solve a 50-node problem easily (40 minutes mean real time). On average, the enhanced triples formulation is 2.31, 21.46, 101.59 and 1,050.75 times faster than the enhanced node-arc formulation for 10-node, 20-node, 30-node, and 40-node respectively in terms of real time solution.

8. REFERENCES

Desrochers, M., Laporte, G. (1991), Improvements and Extensions to the Miller-Tucker-Zemlin Subtour Elimination Constraints, Operations Research Letters, Volume 10, Issue 1, Pages 27-36, https://doi.org/10.1016/0167-6377(91)90083-2.

Dong, Y. (2015), The Stochastic Inventory Routing Problem, Ph.D. Dissertation, Southern Methodist University.

Dong, Y. (2019), private communication.

Dong Y, Bai Y, Olinick EV, Yu AJ (2019) Backhaul profit maximization problem instances. https://scholar.smu.edu/engineering_management_research/2/

Dong, Y., Olinick, E. V., Kratz, T. Jason, Matula, D. W., (2015), A Compact Linear Programming Formulation of the Maximum Concurrent Flow Problem, Networks, Volume 65, Issue 1, Pages 68-87, https://doi.org/10.1002/net.21583.

Dong, Y., Tao, X., Zhou, J. (2006), Optimization of Vehicle Routing and Pricing Model for the Transport Problem with Backhaul, Modern Transportation Technology, Issue 4, Pages 42-45. http://en.cnki.com.cn/Article_en/CJFDTotal-JTJZ200604012.htm.

Fischetti, M., González, J., Toth, P., (1998). Solving the Orienteering Problem Through Branch-and-Cut, INFORMS Journal on Computing, Volume 10, Issue 2, 1998, Pages 133-148. https://doi.org/10.1287/ijoc.10.2.133.

IBM ILOG CPLEX Optimization Studio CPLEX User's Manual Version 12 Release 6, https://www.ibm.com/support/knowledgecenter/SSSA5P_12.6.2/ilog.odms.studio.help/pdf/usrcplex.pdf.

Matula, D. (1986), A New Formulation of the Maximum Concurrent Flow Problem a Proof of the Maximum-Concurrent-Flow/Max-Elongation Duality Theorem, Unpublished, available on line at https://s2.smu.edu/~matula/MCFP86.pdf.

Miller, C., Tucker, A., Zemlin, R. (1960), Integer Programming Formulation of Traveling Salesman Problems, Journal of the ACM, Volume 7, Issue 4, Pages 326-329, https://doi.org/10.1145/321043.321046.

Open Math Reference, Ellipse, available on line at https://www.mathopenref.com/ellipse.html.

Sherali, H., Smich, J. C., (2001), Improving Discrete Model Representations via Symmetry Considerations, Management Science, Volume 47, Issue 10, 2001, Pages 1396-1407, https://doi.org/10.1287/mnsc.47.10.1396.10265

Yu, J., Dong, Y., (2013), Maximizing Profit for Vehicle Routing under Time and Weight Constraints, International Journal of Production Economics, 145, 2, Pages 573-583, https://doi.org/10.1016/j.ijpe.2013.05.009.