Endüstri Mühendisliği Dergisi
Cilt: 29 Sayı: 1-2 Sayfa: (15-30)

Makale

EXACT SOLUTION APPROACHES FOR THE DIRECTED BI-OBJECTIVE CHINESE POSTMAN PROBLEM

Ezgi Eroğlu[1], Meral Azizoğlu[2]*

[1]Middle East Technical University, Ankara 06800, Turkey
e-mail: ezgieroglu@metu.edu.tr

[2]Department of Industrial Engineering
Middle East Technical University, Ankara 06800, Turkey
e-mail: ma@metu.edu.tr

Geliş Tarihi:18.05.2018; Kabul Ediliş Tarihi: 17.09.2018

ABSTRACT

In this study, we consider a directed bi-objective Chinese Postman Problem with two additive objectives (like total cost and total distance) and propose two solution approaches to generate all non-dominated objective vectors. The first approach, namely classical approach, uses the optimal solutions of the mixed integer linear programs and generates the non-dominated objective vectors' set sequentially. The second approach, namely branch and bound algorithm takes its spirit from the optimal solutions of the linear programming relaxations and generates the non-dominated objective vectors' set simultaneously. The results of our extensive computational study show that our approaches are capable of solving large-sized problem instances in reasonable times.

Keywords: Bi-objective Programming, Chinese Postman Problem, Mixed Integer Linear Programming, Classical Approach, Branch and Bound Algorithm

YÖNLÜ İKİ OBJEKTİFLİ ÇİNLİ POSTACI PROBLEMİ İÇİN KESİN ÇÖZÜM YAKLAŞIMLARI

ÖZ

Bu çalışmadaikitoplamsalkriterli (toplammaliyetvetoplammesafegibi) yönlüçinlipostaciproblemielealınmışvetümbastırılamayanobjektifvektörleriniyaratmakiçinikiçözümyaklaşımıgeliştirilmiştir. Birinciyaklaşım,klasikyaklaşım, karmaşıkkesiklidoğrusalprogramların optimal çözümlerinikullanmaktavebastırılamayanobjektifvektörsetiniseriolarakyaratmaktadır. Ikinciyaklaşım, dal vesınıralgoritması, doğrusalprogramlamagevşetimlerinin optimal çözümlerinikullanmaktavebastırılamayanobjektifvektörsetindekiçözümleriaynıandayaratmaktadır.Deneyselçalışmamızınsonuçlarıyaklaşımlarımızınbüyükboyutluproblemlerimakulsürelerdeçözdüğünügöstermektedir.

Anahtar Kelimeler: İki-objektifli programlama, çinli postacı problemi, karmaşık tam sayılı doğrusal programlama, klasik yaklaşım, dal ve sınır algoritması

* Corresponding Author

INTRODUCTION

The arc routing problems entail determining a minimum-cost traversal of a specified subset of arcs on a graph. The earliest study on the arc routing problems is the Königsberg Bridge Problem that is concerned with the existence and determination of a closed walk traversing each of the seven bridges once and exactly once. The bridges of Königsberg are represented as arcs and each arc joins a pair of the four junction points (nodes) that can be considered as islands and shores. The Königsberg Bridge Problem was solved by the Swiss mathematician Leonhard Euler in 1736. The Chinese Postman Problem (CPP) deals with the situations where a closed walk does not exist and determines a minimum length walk covering each arc at least once. The CPP was initially posed by Guan (1962) and stated as follows: Leaving from his post office, a postman needs to visit the households on each block in his route, delivering and collecting letters, and then returns to the post office. He would like to cover this route by traveling the minimum possible distance.

A wide range of applications relevant to transportation, urban planning and manufacturing have been mentioned in the literature.

Some noteworthy transportation and urban planning applications of the CPP addressed in the literature and are of interest for many countries include postal delivery, parcel services, garbage collection, milk delivery, snow clearance, street sweeping services, parking meter coins collection, meter reading, routing of salt trucks and inspection of streets for maintenance and airline scheduling.

Amine and Djellab (2013) show the meaningfulness of the CPP concepts in a wide range of manufacturing applications. Dewill et al. (2011) and Han & Na (1999) discuss the laser or water cutting technology that use water or laser to cut materials and follow Chinese walks for efficiency concerns.

Amine and Djellab (2013) discuss the use of CPP walks in programming automatic arm in the surface painting, like in cars manufacturing, especially in automotive car body painting. The task consists of using an atomizer over a painting surface using a back and forth movement so called CPP walk, above materials.

Manber & Israni (1984) introduce the Cutting Path Determination Problem to solve a flame torch paths problem of material cutting. The problem was modeled as a dynamic rural postman problem, where the graph can be changed each time a piece is cut out. Imahori et al. (2008) and Rodrigues & Ferreira (2012) study the cutting hard materials addressing both packing and cutting path problems. They note that their problem motivates an arc routing modeling as the cutter head requires more time to cut each piece.

The classical CPP assumes that each arc is represented by a single weight and the objective is to determine the tour with the minimum total weight. The bi-objective CPPs assume that each arc is defined by two weights, like cost and distance, distance and priority, cost and time. Many practical implications might require consideration of two weights, thereby two objectives. For example, the total distance of the route should be minimized subject to the constraint that its total cost is below a threshold value. Some other concern might be minimizing the total cost subject to the constraint the total travel distance is below a specified level. Moreover, the decision maker may like to see the solutions over which he/she could make trade-offs among two objectives. For example one may like to see the amount of compromise that should be made on the total distance to reduce the total cost to some specified level.

Despite its practical importance, the literature on the bi-objective CPP is quite scarce. To the best of our knowledge, there is a unique study by Prakash et al. (2009) that is limited to the complete enumeration of all feasible solutions among which the non-dominated ones are selected. Recognizing this important gap in the literature, we study a bi-objective CPP with two additive objectives (total cost and total distance) and propose two solution approaches: a classical approach and a branch and bound algorithm. The classical approach, uses the optimal solutions of the mixed integer linear programs. The branch and bound algorithm generates all non-dominated (efficient) points and benefits from the optimal solutions of the Linear Programming Relaxation (LPR) to define our branching scheme and lower and upper bounds.

The rest of the paper is organized as follows: In Section 2, we review the related literature. Section 3 defines our problem and presents related mathematical models. In Section 4, we first define the efficient solutions together with their properties and then state the classical approach that sequentially generates all nondominated objective vectors. In Section 5 we first settle the complexity of the problem and discuss the branch and bound algorithm that simultaneously generates all nondominated objective vectors. Section 6 discusses the results of our extensive computational experiment and Section 7 concludes the study.

2. BASICS AND THE RELATED RESEARCH

The CPP is an arc routing problem in which a single postman serves a number of streets from a post office. The classical CPP finds the minimum cost tour passing through every arc of graph at least once. This problem is firstly studied by Guan (1962). To find minimum total cost tour, the concept of unicursality is used by Ford and Fulkerson (1962). A directed graph is said to be unicursal (Eulerian) if for each node, the number of entering arcs is equal to the number of leaving arcs. An undirected graph is said to be universal if even number of arcs are incident to each node. To make a graph Eulerian a minimum weight perfect matching is defined and optimal solution to the CPP is found. The problem is solved by a polynomial time algorithm developed by Edmonds and Johnson (1973).

Eiselt et al. (1995) give two mathematical models for the undirected CPP. The first model is based on a perfect matching idea that converts a nonunicursal graph into a unicursal one by using the shortest paths. The second model, on the other hand, is based on Edmonds' blossom inequalities (see Edmonds, 1963) and it depends on the density of arcs.

In the directed case, the problem has a feasible solution provided that the graph is strongly connected, i.e., there exists a directed path between every pair of nodes. The optimal solution is found by a polynomial time algorithm as proposed in Eiselt et al (1995). It is shown that a minimum cost unicursal graph can be constructed using the transportation model where the decision variables represent the number of times each arc has to be traversed.

Windy graphs are undirected graphs in which the cost of traversing an arc depends on the direction of travel. Windy postman problem (WPP) was firstly introduced by Minieka (1979). Brucker (1981) shows that the WPP is strongly NP-hard and Win (1989) shows that the problem is polynomially solvable when the graph is unicursal. Grötschel and Win (1992) describe the cutting plane algorithm to solve the WPP.

Mixed graphs contain both undirected and directed arcs and consist of determining a least-cost traversal through every arc of the graph at least once. Papadimitriou (1976) shows that finding the minimum cost solution to the mixed CPP is strongly NP-hard and Edmonds and Johnson (1973) show that the problem is polynomially solvable when the graph is unicursal. To find the optimal solution, some authors, including Christofides et al. (1984) and Nobert and Picard (1991), have used integer linear programming formulations. The branch and cut algorithm is proposed by Corberan et al. (2012) to solve large size Mixed CPP instances.

Prakash et al. (2009) consider two additive objectives for the CPP and presents a heuristic procedure that finds a set of feasible solutions giving higher priority to the first objective. The performance of Prakash et al. (2009)'s procedure depends on the choice of the priorities and the procedure may return a single nondominated point or a subset of all nondominated points or a set of all nondominated points.

The biobjective arc routing problems with multi postmen have also been studied in the last decade. Lacomme et al. (2006), Mei et al. (2011) and Grandinetti et al. (2012) propose heuristic approaches that return the approximate set of nondominated points with respect to the total cost and maximum cost criteria.

In this study, we propose two approaches each returns the set of all nondominated points for the CPP. To the best of our knowledge, our study is the first attempt for the exact solution of the bi-objective arc routing problems.

3. PROBLEM DEFINITION AND THE RELATED MODELS

Consider a directed graph G = (N, A) that consists of a set of A arcs and a set of N nodes. Arc *(i,j)* establishes a connection between nodes *i* and *j* and is characterized by two weights, c_{ij} and d_{ij}.*cij* may stand for the cost of traversing arc (*i,j*) whereas d_{ij} is its travel time or distance. The main decision of the problem is explained via the variable X_{ij} as follows:

X_{ij}=number of times arc *(i,j)* is traversed

The objective of the classical CPP is to minimize the total cost that is expressed as

$$\text{Minimize} \sum_{(i,j)\in A} c_{ij} X_{ij} \qquad (1)$$

The constraint sets are stated below:

i. Each arc should be traversed at least once:

$$X_{ij} \geq 1 \text{and integer} \quad \forall (i,j) \in A \qquad (2)$$

ii. The flow to a node should be conserved that is for each entering arc, there should be a departing counterpart.

$$\sum_{j \mid (i,j)\in A} X_{ij} = \sum_{j \mid (j,i)\in A} X_{ji} \qquad \forall i \in N \qquad (3)$$

We simply refer to the above CPP model that minimizes (1) while satisfying (2) and (3) as P_C.P_D is the another CPP model that minimizes (4) below while satisfying (2) and (3).

$$\text{Minimize} \sum_{(i,j)\in A} d_{ij} X_{ij} \qquad (4)$$

Solution Procedures for the classical CPP

Eiselt et al (1995) show that the classical CPP can be represented by a transportation model, hence can be solved in polynomial time. The decision variable set of the transportation model, X_{ij} is as defined below:

X_{ij} = number of times the shortest path between (*i,j*) is traversed

Thetransportation model uses the following definitions:

i. $d_i^+ = \begin{cases}\text{number of incoming arcs to node i} \\ \text{indegree of node i}\end{cases}$

$d_i^- = \begin{cases}\text{number of outgoing arcs from node i} \\ \text{outdegree of node i}\end{cases}$

Node *i*is called balanced if $d_i^+ = d_i^-$.

ii. The sets below are defined for the unbalanced nodes:

I= Set of supply nodes = $\{i \mid d_i^+ > d_i^-\}$

J = Set of demand nodes = $\{i \mid d_i^+ < d_i^-\}$

iii. $S_i = d_i^+ - d_i^- \qquad \forall i \in I$

$D_j = d_j^- - d_j^+ \qquad \forall j \in J$

iv. SP_{ij} = length of the shortest path between nodes *i* and *j*

$\forall i \in I, \quad \forall j \in J$

The model is as stated below:

$$\text{Minimize} \sum_{i \in I} \sum_{j \in J} SP_{ij} X_{ij} + \sum_{(i,j)\in A} c_{ij}$$

subject to

$$\sum_{j \in J} X_{ij} = S_i \qquad \forall i \in I$$

$$\sum_{i \in I} X_{ij} = D_j \qquad \forall j \in J$$

$$X_{ij} \geq 0 \text{ and integer} \qquad \forall i \in I, \ j \in J$$

An optimal objective function value of the above transportation model gives the optimal total cost of the classical CPP.

The Constrained Chinese Postman Problem

Consider the following constraint that imposes an upper bound *k* on the second criterion, i.e., total distance travelled:

$$\sum_{(i,j)\in A} d_{ij} X_{ij} \leq k \qquad (5)$$

Minimizing (1) subject to the constraint sets (2), (3) and (5) is a single constrained CPP model. We hereafter refer to the single constrained CPP as P_C,*k*.

Figure 1 illustrates the feasible and optimal solutions of the P_C and P_C,*k* problems via a 25 nodes and 43 arcs instance taken from Malandraki and Daskin (1993).

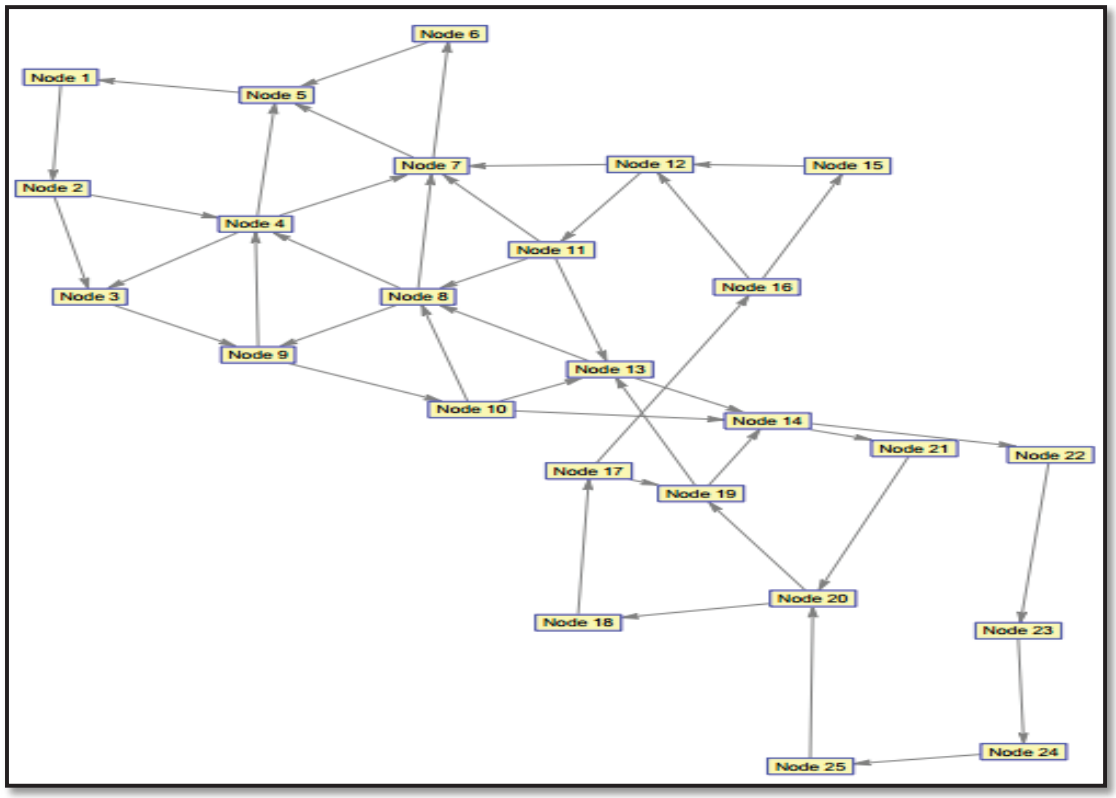

Figure 1. The Example Instance

The arc weights, i.e. (c_{ij}, d_{ij}) values are given in Table 1 below.

Table 1. The Arc Weights of the Example Instance

(i, j)	(c_{ij}, d_{ij})	(i, j)	(c_{ij}, d_{ij})	(i, j)	(c_{ij}, d_{ij})	(i, j)	(c_{ij}, d_{ij})
(1,2)	(40,40)	(8,4)	(50,50)	(12,7)	(90,90)	(18,17)	(30,30)
(2,3)	(30,30)	(8,7)	(30,30)	(12,11)	(20,20)	(19,13)	(40,40)
(2,4)	(40,12)	(8,9)	(60,60)	(13,8)	(70,70)	(19,14)	(70,70)
(3,9)	(40,40)	(9,4)	(70,70)	(13,14)	(70,21)	(20,18)	(30,30)
(4,3)	(40,12)	(9,10)	(60,60)	(14,21)	(20,80)	(20,19)	(30,30)
(4,5)	(30,30)	(10,8)	(60,60)	(14,22)	(40,40)	(21,20)	(20,70)
(4,7)	(50,50)	(10,13)	(60,9)	(15,12)	(40,20)	(22,23)	(30,30)
(5,1)	(50,50)	(10,14)	(30,36)	(16,12)	(40,60)	(23,24)	(30,30)
(6,5)	(50,15)	(11,7)	(80,80)	(16,15)	(40,20)	(24,25)	(80,20)
(7,5)	(50,50)	(11,8)	(70,70)	(17,16)	(40,40)	(25,20)	(70,20)
(7,6)	(30,9)	(11,13)	(30,30)	(17,19)	(30,30)		

The optimal solution to the P_C problem is stated below:

$X^*_{1,2} = 5,\ X^*_{2,3} = 4,\ X^*_{5,1} = 5,\ X^*_{7,5} = 3,\ X^*_{3,9} = 5, X^*_{9,10} = 5,\ X^*_{12,11} = 3,\ X^*_{16,12} = 3,$

$X^*_{10,14} = 3,\ X^*_{13,14} = 2,\ X^*_{17,16} = 4,\ X^*_{18,17} = 5,\ X^*_{20,18} = 5,\ X^*_{21,20} = 5,\ X^*_{14,21} = 5$

and all other $X^*_{i,j}$ values are equal to 1.

The optimal total cost $Z_C^* = \sum_{(i,j)\in A} c_{ij} X_{ij}^*$

$$Z_C^* = 40 \times 5 + 30 \times 4 + 40 \times 1 + \cdots + 30 \times 1 + 80 \times 1 + 70 \times 1 = 3700$$

The optimal solution to the $P_D$problem is stated below:

$X^*_{1,2} = 5,\ X^*_{2,4} = 4,\ X^*_{4,3} = 4,\ X^*_{5,1} = 5,\ X^*_{6,5} = 3,\ X^*_{7,6} = 3,\ X^*_{3,9} = 5, X^*_{9,10} = 5,\ X^*_{12,11} = 3,\ X^*_{15,12} = 3,\ X^*_{16,15} = 3,\ X^*_{10,13} = 3,\ X^*_{13,14} = 4,\ X^*_{17,16} = 4,\ X^*_{18,17} = 5,\ X^*_{20,18} = 5,\ X^*_{22,23} = 5,\ X^*_{14,22} = 5,$
$X^*_{23,24} = 5,\ X^*_{24,25} = 5,\ X^*_{25,20} = 5$

and all other $X^*_{i,j} = 1$.

The optimal total distance $Z_D^* = \sum_{(i,j)\in A} d_{ij} X_{ij}^*$

$$Z_D^* = 40 \times 5 + 30 \times 1 + \cdots + 30 \times 5 + 20 \times 5 + 20 \times 5 = 3755$$

Now assume *k* is 3890.

The constraint $\sum_{(i,j)\in A} d_{ij} X_{ij}^* \leq 3890$ is not satisfied since $\sum_{(i,j)\in A} d_{ij} X_{ij}^* = 3917$.

The (P_C, k) problem with *k* value of 3890 returns the following optimal solution:

$X^*_{1,2} = 5,\ X^*_{2,3} = 4,\ X^*_{5,1} = 5,\ X^*_{6,5} = 3,\ X^*_{7,6} = 3,\ X^*_{3,9} = 5,\ X^*_{9,10} = 5,\ X^*_{12,11} = 3,\ X^*_{16,12} = 3,$
$X^*_{10,14} = 3,\ X^*_{13,14} = 2,\ X^*_{17,16} = 4,\ X^*_{18,17} = 5,\ X^*_{20,18} = 5,\ X^*_{21,20} = 5,\ X^*_{14,21} = 5$ where all other $X^*_{i,j}$ values are equal to 1.

The optimal total cost, which is denoted by $(Z_C, k)^*$, is equal to:

$$Z_{C,k}^* = \sum_{(i,j)\in A} c_{ij} X_{ij}^* = 40 \times 5 + 30 \times 4 + 40 \times 1 + \cdots + 30 \times 1 + 80 \times 1 + 70 = 3760$$

The travel distance constraint is also satisfied:

$$\sum_{(i,j)\in A} d_{ij} X_{ij}^* = 3865 < 3890$$

Note that $(Z_C, k)^*$ is bigger than Z_C^*, i.e.,$Z_C^* = 3700 < 3760 = Z_{C,k}^*$

The steps our study can be summarized as follows: We first define the properties of the efficient solutions and use them in our mixed integer linear model and its linear programming relaxation. We use mixed integer linear model in classical approach for sequential generation of all nondominated objective vectors. We use the linear programming relaxation of the model in branch and bound algorithm for simultaneous generation of all nondominated objective vectors. The final step of the study evaluates the performances of the classical approach and branch and bound algorithm.

4. NONDOMINATED OBJECTIVE VECTORS AND THE CLASSICAL APPROACH

A solution u is called efficient if there is no other solution v, having $Z_C^v \leq Z_C^u$ and $Z_D^v \leq Z_D^u$ with strict inequality holding at least once. The resulting objective function vector (Z_C^u, Z_D^u) is said to be nondominated and we say objective vector (Z_C^v, Z_D^v) dominates the objective vector (Z_C^v, Z_D^v).

In subsection 4.1 we define the properties of the efficient solutions. In subsection 4.2 we define a way to find a single nondominated objective vector and subsection 4.3 we discuss the procedure that finds all nondominated objective vectors.

4.1 Properties of the Efficient Solutions

We present three properties of the efficient solutions.

Property 1. In all efficient solutions, for any node i whose indegree is 1, i.e.,$d_i^+ = 1$, $x_{ji} \geq d_i^-$ where arc (j, i) is incident to node i.

Proof. There are at least d_i^- inflows to node i; hence, at least d_i^- outflows from node i. The only way of realizing outflows is traversing arc (j, i), at least d_i^- times. It follows that in all feasible, hence efficient, solutions, $x_{ji} \geq d_i^-$.

Property 2. In all efficient solutions, for any node i whose outdegree is 1, i.e. d_i^-=1, $x_{ij} \geq d_i^+$ for the arc (i, j) incident to node i.

Proof. Similar to that of Property 1; hence, it is omitted.

Property 3. In all efficient solutions, an arc incident to node r is traversed exactly once if node r is not on any shortest path that connects $i \in I$ to $j \in J$.

Proof. An optimal solution to the transportation problem (discussed in Section 3) that is equivalent to the P_C problem (P_D problem) with arc weights c_{ij} (arc weights d_{ij}) traverses any incident arc to a balanced node r exactly once. This follows that if any balanced node is not on any shortest path between $i \in I$ to $j \in J$, relative to botharc weights, i.e., c_{ij} *and* d_{ij} then it is traversed exactly once in all efficient solutions.

Using the result of Property 3, if node r is not on any shortest path between $i \in I$ to $j \in J$, relative to both arc weights we set the flows of all arcs incident to node r to 1, and call node r as eliminated node. We define

$$A_{ir} = \{i \mid arc\ (r, i)\ exists\}$$

$$B_{jr} = \{j \mid arc\ (j, r)\ exists\}$$

We thereafter refer set E as the set of eliminated nodes and redefine N as N / E.

As $X_{j,r} = X_{r,i}$ for all (j, r) and (r, i) the flow conservation relations are rewritten as:

$$|A_{ir}| + \sum_j X_{i,j} = \sum_j X_{j,i} + |B_{jr}|$$

$\forall\, r \in E,\ \forall\, i, j$ incident to r

Using the results of Property 1 and Property 2, the following constraints are added.

$$X_{ij} \geq d_i^+ \qquad \forall\, i \mid d_i^- = 1$$

$$X_{ij} \geq d_i^- \qquad \forall\, i \mid d_i^+ = 1$$

4.2 Finding A Nondominated Objective Vector

Consider the following modified P_c model.

$$\text{Minimize} \sum_{(i,j)\in A} c_{ij} X_{ij} \tag{5}$$

subject to

$$X_{ij} \geq 1 \tag{6}$$

$$X_{ij} \geq d_i^+ \qquad \forall\, i \mid d_i^- = 1 \tag{7}$$

$$X_{ij} \geq d_i^- \qquad \forall\, i \mid d_i^+ = 1 \tag{8}$$

$$\sum_j X_{ij} = \sum_j X_{ji} \qquad \forall\, i \in N \tag{9}$$

$$|A_{ir}| + \sum_{j} X_{i,j} = \sum_{j} X_{j,i} + |B_{jr}| \quad \forall r \in E,\ \forall i,j \text{incident to} r \tag{10}$$

We hereafter refer to constraint sets (6) through (10) as $x \in X$.

Let Z_C^* be the its optimal Z_C value. Z_C^* is a valid lower bound on the Z_C values of all efficient solutions. Any optimal solution to the P_C problem may not be efficient since there may exist alternative optimal solutions having smaller Z_D values.

Among the alternative optimal solutions to the P_C problem, the one having the smallest Z_D value requires the solution of the following problem.

$$Minimize \sum_{(i,j)\in A} d_{ij} X_{ij}$$

subject to $x \in X$

$$\sum_{(i,j)\in A} c_{ij} X_{ij} = Z_C^*$$

We hereafter refer to the above model as (P_D, Z_C^*). Note that (P_D, Z_C^*) requires solving the P_C problem. In place of solving the P_C problem and then the (P_D, Z_C^*) problem, one may solve the following problem:

$$Minimize \left(\sum_{(i,j)\in A} c_{ij} X_{ij} \right) + \in_D \left(\sum_{(i,j)\in A} d_{ij} X_{ij} \right)$$

subject to $x \in X$

The value of $\in_D$ should be small enough that the smallest $\sum_{(i,j)\in A} c_{ij} X_{ij}$ value should not increase even for the largest possible value of $\sum_{(i,j)\in A} d_{ij} X_{ij}$,. Accordingly,

$$Z_C^* + \in_D D_{max} \leq Z_C^* + 1 + \in_D D_{min} \tag{11}$$

where D_{min} and D_{max} are the smallest and largest possible value of the total distance, respectively. The inequality (11) follows that

$$\in_D (D_{max} - D_{min}) \leq 1$$

$$\in_D \leq \frac{1}{(D_{max} - D_{min})}$$

D_{min} is the optimal Z_D value of the P_D problem. D_{max} is an upper bound on the Z_D value of all efficient solutions; hence, it is the $(\sum_{(i,j)\in A} d_{ij} X_{ij})$ value of the (P_C) problem. In other words, first we solve the P_C problem and obtain optimal X_{ij} values, then by using those X_{ij} values in equation of $(\sum_{(i,j)\in A} d_{ij} X_{ij})$, we obtain the D_{max} value.

In our experiments, we set $\in_D$ to $\frac{1}{(D_{max} - D_{min})+1}$

4.3 Finding the Set of All Nondominated Objective Vectors

An optimal solution to the following constrained problem, P_C, k, gives a nondominated objective vector and a corresponding efficient solution (see Haimes et al., 1971).

$$\text{Minimize} Z_C + \in_D Z_D$$

subject to $x \in X$

$$Z_D \leq k$$

This follows, all nondominated objective vectors and corresponding efficient solutions can be generated by solving (P_C, k) for all k in $[D_{min}, D_{max}]$. The following procedure systematically varies the value of k and returns the set of all nondominated objective vectors.

Procedure 1. The Classical Approach (CA)

Step 0. Solve (P_C) Minimize Z_C subject to $x \in X$

Let $D_{\max}$ be the Z_D value of the P_C

Solve (P_D) Minimize Z_D subject to $x \in X$

Let $D_{\min}$ be the Z_D value of the P_D

Let $\in_D = \frac{1}{(D_{max} - D_{min})+1}$

Step 1. Solve(P_C, k) Minimize $Z_C + \in_D Z_D$

subject to $x \in X$

$Z_D \leq k$

Let (Z_C^*, Z_D^*) be the solution.

Set $r = r + 1$

Step 2. If $Z_D^* = D_{\min}$ then stop, all r nondominated objective vectors are generated.

Else set $k = Z_D^* - 1$ and go to Step 1.

Note that each iteration Procedure 1 generates a nondominated objective vector. When the procedure terminates, the set of all nondominated objective vectors are generated.

The number of nondominated objective vectors, r, is at most D_{max} - D_{min} + 1. Hence, the Procedure 1 iterates pseudo polynomial number of times.

5. COMPLEXITY AND THE BRANCH AND BOUND ALGORITHM

In this section, we first settle the complexity of the constrained CPP, i.e., P_C, k and then present a Branch and Bound algorithm for its exact solution.

Theorem 1. The constrained CPP is strongly NP-hard.

Proof. Recall that the CPP reduces to the following 'Transportation Problem' (see Section 3).

$$\text{Minimize} \sum_{i \in I} \sum_{j \in J} SP_{ij} X_{ij} + \sum_{(i,j) \in A} c_{ij}$$

$$\sum_{j \in J} X_{ij} = S_i \qquad \forall i \in I$$

$$\sum_{i \in I} X_{ij} = D_j \qquad \forall j \in J$$

$$X_{ij} \geq 0 \text{ and integer} \qquad \forall i \in I,\ j \in J$$

Hence, our constrained CPP problem is equivalent to a 'Capacitated Transportation Problem' with the following capacity constraint.

$$\sum_{i \in I} \sum_{j \in J} d_{ij} X_{ij} \leq \bar{D} \text{ where} \bar{D} = k - \sum_{(i,j) \in A} d_{ij}$$

When all supply and demand amounts are '1', the 'Capacitated Transportation Problem' reduces to the Generalized Assignment Problem (GAP). It follows that the P_C, k problem reduces to the GAP. The GAP is NP-hard in the strong sense (see, Martello and Toth, 1990), so is the P_C, k problem.

The problem of generating a single nondominated objective vector is strongly NP-hard as the constrained CPP is strongly NP-hard. It follows that the problem of generating all nondominated objective vectors with respect to two objectives is strongly NP-hard.

Attributing to the complexity of the problem, Procedure 1 that generates all nondominated objective vectors is likely to fall into computational burden. To dispel the burden to some extent, we present an implicit enumeration technique –a Branch and Bound (BAB) algorithm. Our aim is to attain optimal solutions to the large sized instances in reasonable solution times.

Our BAB algorithm generates one nondominated objective vector at a time. It starts with an initial upper bound (incumbent solution) and updates it whenever a complete solution with a better objective function value is reached.

We solve the Linear Programming Relaxation (LPR), i.e., first relax the integrality constraints on the X_{ij} values and then solve the resulting problem to optimality. We benefit from the optimal solution of the LPR to find an initial upper bound on the Z_C value. Below is the stepwise description of our upper bounding procedure.

Procedure 2 - Finding an Initial Upper Bound

Step 1. Solve the LPR.

Step 2. Let $c\tau$ be a cycle that resides all fractional variables. Let X_{rs}^L be the smallest fractional variable in $c\tau$.

Step 3. Let $X_{rs}=\lceil X_{rs}^L \rceil$ and update the fractional variables around $c\tau$ so as to preserve the flow conservation relations.

Step 4. If$X_{rs} = \lceil X_{rs}^L \rceil$ gives a feasible solution, then go to Step 5.

Else let $X_{rs} = \lfloor X_{rs}^L \rfloor$ and update the fractional variables around.

Step 5. If all variables are integers, stop.

Else go to Step 2.

At each node of the BAB tree, we solve the LPR and explore the tree using the fractional variables of the optimal LPR solution. We let X_{rs} be the maximum fractional value and generate the following two child nodes:

Child Node I. Add constraint $X_{rs} > \lceil X_{rs}^L \rceil$

Child Node II. Add constraint $X_{rs} \leq \lfloor X_{rs}^L \rfloor$

We use the optimal solution value of the LPR model

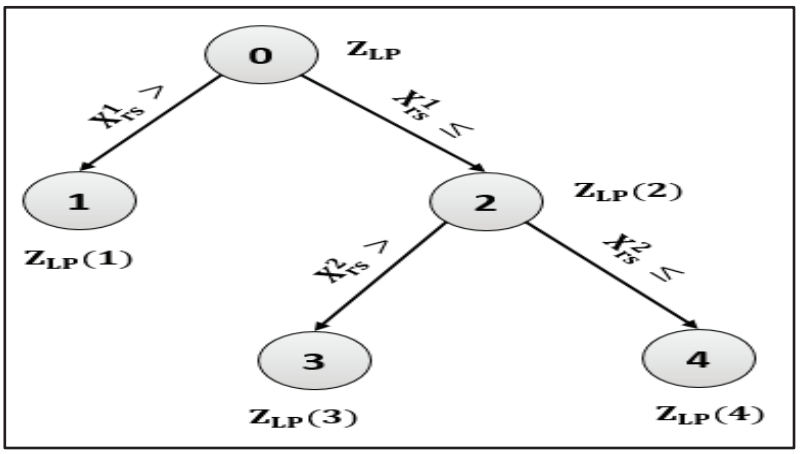

Figure 2. The BAB Tree

as a lower bound. We select the node having the smaller lower bound. For the selected node, we find an upper bound using the upper bounding procedure. Figure 2 illustrates our BAB tree.

Nodes 1 and 2 are generated based on the optimal solution of the LPR solved at Node 0. Node 2 is selected for further branching as $Z_{LP}^2 \leq Z_{LP}^1$. The child nodes of Node 2, i.e., Node 3 and Node 4, use the additional constraints $X_{rs}^2 \geq \lceil X_{rs}^2 \rceil$ and $X_{rs}^2 \leq \lfloor X_{rs}^2 \rfloor$, respectively. The tree explores either from Node 3 if $Z_{LP}^3 \leq Z_{LP}^4$ or from Node 4 if $Z_{LP}^3 > Z_{LP}^4$.

A node is fathomed if one of the following conditions holds:

i. The resulting LPR leads to an infeasible solution.

ii. The resulting LPR gives all integer decision variables. In such a case, the solution

(Z_C^I, Z_D^I) is updated if $Z_C < Z_C^I$ or $Z_C = Z_C^I$ and $Z_D < Z_D^I$.

iii. The objective function value of the resulting LPR is no better than the incumbent solution. That is, $Z_C^{LP} > Z_C^I$ or $Z_C^{LP} = Z_C^I$ and $Z_D^{LP} > Z_D^I$.

We employ a depth first strategy due to its relatively low memory requirements. According to the strategy, we start form the root node and explore branching from the node having smaller Z_C value or smaller Z_D value when the Z_C values are equal.

If both nodes are fathomed, we backtrack to the previous level. We stop when we backtrack to Level 1. The best solution at termination, i.e., incumbent solution is the optimal solution.

We illustrate our branching scheme on example instance I. The optimal solution of the LPR at the initial level is 4211 when k = 3794. The initial upper bound is found as 4400.

The BAB tree of the instance is presented in Figure 3.

The figures on the nodes indicate the order at which they are created. At each node, the lower and upper bound values are reported. At Node 1, the following LPR is solved:

$$Minimize \ Z_C + \in_D Z_D$$
$$Subject\ to \quad x \in X$$
$$X_{14,21} \leq 4$$

The optimal objective function value of the associated LPR is found as $Z_{C,k} = 4250$ for $k = 3794$. We update the upper bound at that node. According to the optimal solution of the LPR, the cycle that resides the fractional variables is given in Figure 4.

We choose the smallest fractional variable in the cycle, i.e., $X_{10,13} = 1.5$. We let $X_{10,13} = 2$ and update the fractional variables around the cycle as shown in Figure 5.

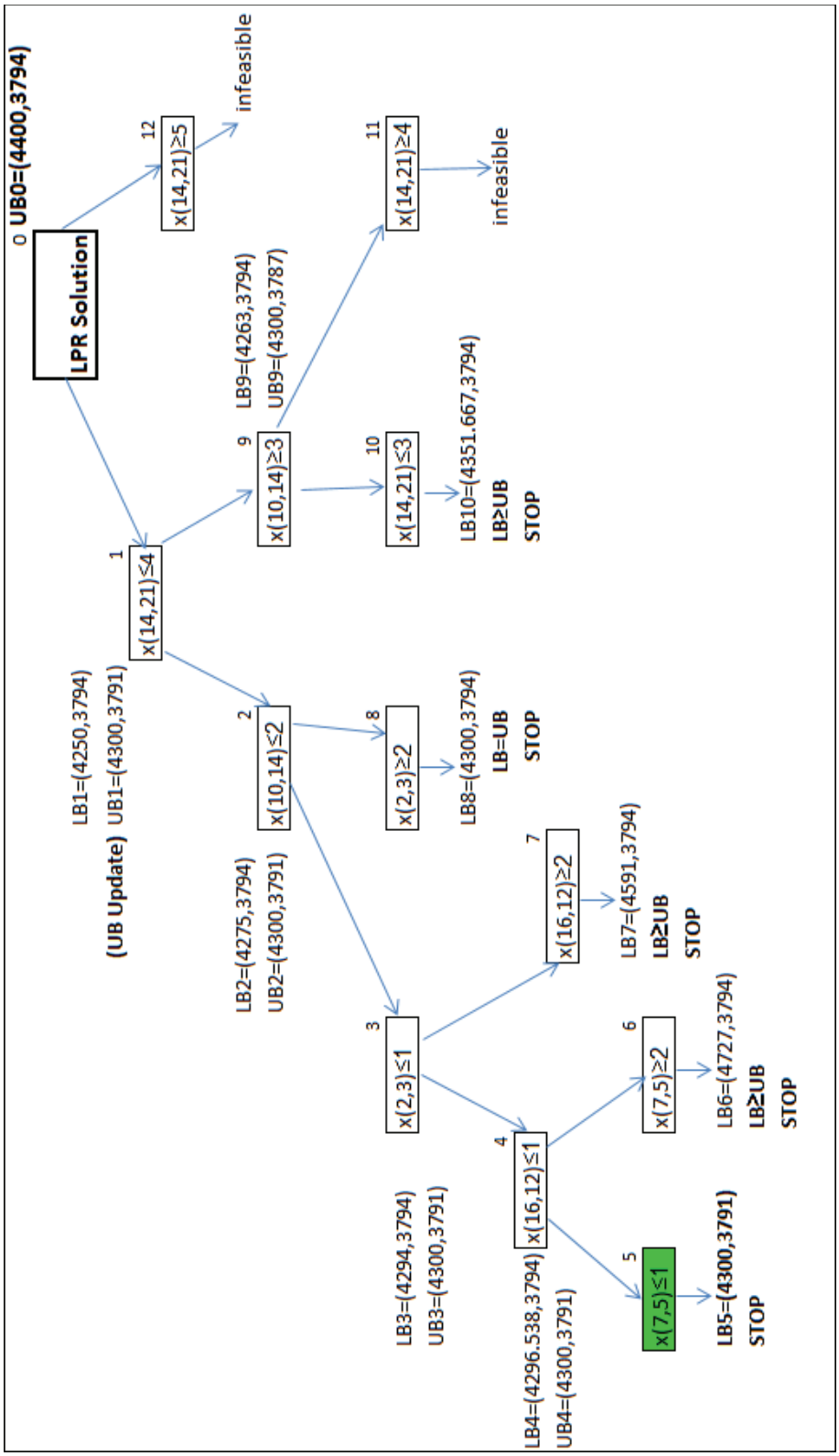

Figure 3. The BAB Tree of the Instance

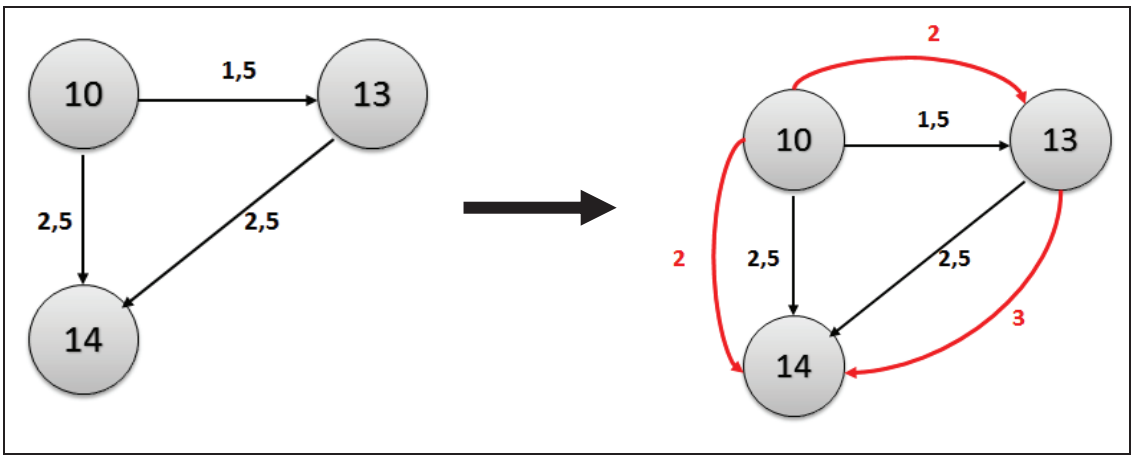

Figure 4. The Illustration of the Upper Bounding Procedure

The upper bounds for the total cost and total distance are found as:

$$UB_C^1 = 4250 + 0.5\,(60 - 30 + 70) = 4300$$

$$UB_D^1 = 3794 + 0.5\,(9 - 36 + 21) = 3791$$

Then, we update the best upper bound from (UB_C, UB_D) = (4400, 3794) to (4300, 3791). At Node 2, we solve the following LPR model:

$$Minimize\ Z_C + \in_D Z_D$$

$$Subject\ to \quad x \in X$$

$$X_{14,21} \leq 4,\ X_{10,14} \leq 2$$

The upper bound in terms of total cost and total distance is then equal to:

$$UB_C^1 = 4250 + 0.5\,(60 - 30 + 70) = 4300$$

$$UB_D^1 = 3794 + 0.5\,(9 - 36 + 21) = 3791$$

Then, we update the upper bound from (UB_C, UB_D) = (4400, 3794) to (4300, 3791).

At Node 2, we solve the following LPR model:

$$Minimize\ Z_C + \in_D Z_D$$

$$Subject\ to \quad x \in X$$

$$X_{14,21} \leq 4,\ X_{10,14} \leq 2$$

The optimal solution of the above LPR is found $Z_{C,k}$= 4275 for k = 3794. The upper bound is not updated as it is no better than the one found in Node 1. According to the solution of the LPR at Node 2, the cycle that resides fractional variables are $X_{2,3}$=1.5, $X_{2,4}$=3.5, $X_{4,3}$=3.5. We choose the smallest fractional variable in the cycle, i.e., $X_{2,3}$ = 1.5. Then, at Node 3, we solve the following LPR model:

$$Minimize\ Z_C + \in_D Z_D$$

$$Subject\ to \quad x \in X$$

$$X_{14,21} \leq 4,\ X_{10,14} \leq 2,\ X_{2,3} \leq 1$$

The optimal solution of the above LPR is found as $Z_{C,k}$ =4294 for k = 3794.

According to the solution of LPR at Node 3, $X_{15,12}$ = 2.85, $X_{16,12}$= 1.15, $X_{16,15}$=2.85 resides fractional cycle. We choose the smallest fractional variable in the cycle, i.e., $X_{16,12}$ = 1.15. Then, at Node 4, we solve the following LPR model:

$$Minimize\ Z_C + \in_D Z_D$$

$$Subject\ to \quad x \in X$$

$$X_{14,21} \leq 4,\ X_{10,14} \leq 2,\ X_{2,3} \leq 1,\ X_{16,12} \leq 1$$

The optimal solution of the LPR is found as $Z_{C,k}$ = 4296,538 for k = 3794. According to the solution of LPR

at Node 4, the cycle that resides fractional variables are $X_{6,5} = 2.885$, $X_{7,5} = 1.115$, $X_{7,6} = 2.885$.

At Node 5, when we solve the following LPR by choosing the smallest fractional variable in the cycle, the optimal solution is equal to the updated upper bound value at Node 1, i.e., $(LB_C, LB_D) = (4300, 3791)$. Since the resulting LPR gives all integer decision variables, we stop branching and continue to explore from the other node at the same level.

$$Minimize\ Z_C + \in_D Z_D$$

$$Subject\ to\quad x \in X$$

$$X_{14,21} \leq 4,\ X_{10,14} \leq 2,\ X_{2,3} \leq 1,\ X_{16,12} \leq 1,\ X_{7,5} \leq 1$$

Nodes 6, 7, 8 and 10 are fathomed since the objective functions of the resulting LPRs are greater than or equal to those of the incumbent solution. Nodes 11 and 12 are also fathomed as their since the resulting LPRs leads to an infeasible solution. The best solution at termination is optimal and it is found at Node 5.

6. COMPUTATIONAL EXPERIMENT

We design an experiment to evaluate the performances of our algorithms. The number of nodes and number of arcs used in our experiment are tabulated in Table 2.

Table 2. The (N, M) Values

N	100	100	300	300	300	500	500
M	200	400	600	675	750	1200	1300

The networks with 100 nodes are from http://www.ing.unibs.it/~orgroup/instances.html. The networks with 300 and 500 nodes are from http://www.uv.es/corberan/instancias.html. We generate the arc weights (c_{ij} and d_{ij} values) from discrete uniform distributions in [1, 100].

All experiments are carried out on an Intel(R) Core(TM) i5-3317U and clocked at 1.70 GHz with 4 GB RAM. The BAB algorithm is coded in Java, Eclipse Luna version 4.4.0.

For each (N, M) combination, we generate 10 problem instances. Hence, our experiment resides 60 problem instances. For each instance, we generate the set of all nondominated objective vectors; hence solve many combinatorial optimization problems.

We set a termination limit of 3600 seconds for the BAB algorithm and CA.

Table 3. The Number of the Nondominated Objective Vectors

		Number of nondominated objective vectors	
N	**M**	**Average**	**Maximum**
100	200	43	150
100	400	237	425
300	600	106	152
300	750	426	904
500	1200	162	294
500	1300	355	503

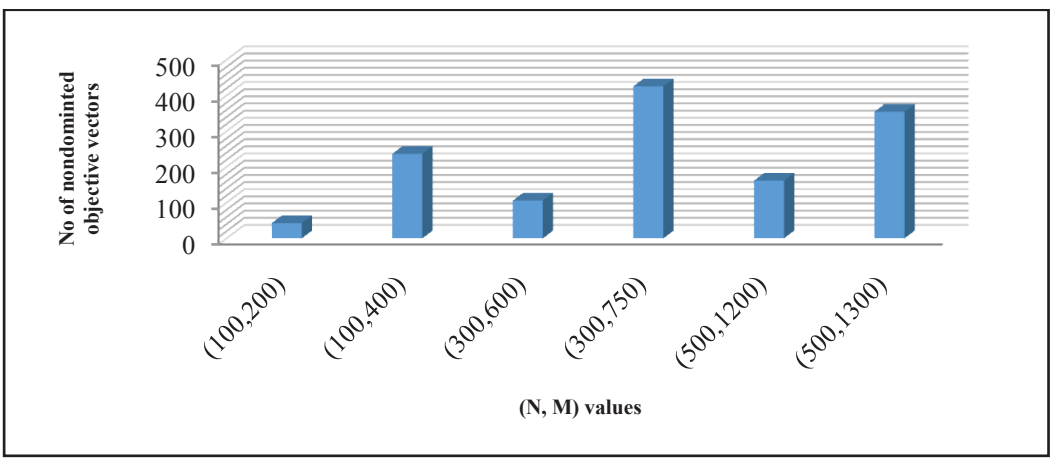

Figure 5. The Average Number of the Nondominated Objective Vectors

The number of nondominated objective vectors is reported in Table 3. The table includes the average and maximum number of nondominated objective vectors for each problem combination.

The average number of the nondominated objective vectors for each combination is also represented in Figure 5. The figure shows the effect of the problem size parameters on the number of the nondominated objective vectors.

Note from Figure 5 that the number of the nondominated objective vectors is highly dependent on the problem size. Note from the figure that for N = 100, the average number of the nondominated objective vectors increases from 43 to 237 as M increases from 200 to 400. For N = 300, these increases are from 106 to 426 when M increases from 600 to 750. For N = 500, the average CPU times increases more than two times when M increases by 100.

Table 4 reports the average and maximum CPU times in seconds and the number of nodes generated BAB algorithm in generating a nondominated objective vector.

We observe from Table 4 that as N or M increases, the CPU time to find a nondominated objective vector increases significantly. Finding a nondominated objective vector takes 0.382 seconds on average for the networks with 100 nodes and 200 arcs, while it reaches to 1.854 seconds for the network with (N, M) values of (500, 1200). When N = 100 and M = 200, the average number of nodes generated is 16 and the average CPU time to find a nondominated objective vector is 0.382 seconds. For N = 100, when M increases to 400, the average number of nodes and CPU time rise to 141 and 1.243 seconds, respectively. The increases in the complexity of the solutions with increases in N or M can be attributed to the increases in the complexity of the LPR models.

Table 4. The Performance of the BAB and CA-per Nondominated Objective Vector

		CA CPU Time		BAB CPU Time		BAB Nodes	
N	M	Average	Maximum	Average	Maximum	Average	Maximum
100	200	0.298	0.490	0.382	1.069	16	27
100	400	0.287	0.311	1.243	1.613	141	197
300	600	2.593	3.115	1.01	1.319	70	96
300	750	2.292	2.993	2.779	3.765	177	235
500	1200	5.183	5.727	1.854	2.431	187	384
500	1300	5.286	5.696	4.224	5.563	152	202

Table 5. The Performance of the BAB and CA-per Nondominated Objective Vector

		CA CPU Time		BAB CPU Time		BAB Nodes	
N	M	Average	Maximum	Average	Maximum	Average	Maximum
100	200	11.448	33.43	8.543	19.097	848	3138
100	400	67.179	118.23	306.863	634.37	35030	70660
300	600	267.643	352.9	107.245	182.76	7505	13478
300	750	1017.37	2210.95	1254.83	2812.6	79855	175564
500	1200	833.533	1506.47	306.508	561.688	24377	32992
500	1300	1879.3	2604.95	1528.88	2716.86	54747	93794

Table 5 reports on the performance of generating the set of nondominated objective vectors.

Note from Table 5 that when N or M increases, the number of nondominated objective vectors and time to generate a single nondominated objective vector increase significantly. These in turn, increase the time to generate the set of all nondominated objective vectors. For the (N, M) values of (100, 400), (300, 750) and (500, 1300), the average CPU times are about 307, 1255 and 1539 seconds, respectively. The CPU times are almost tripled when the problem sizes are increased from (100, 400) to (500, 1300).

Tables 4 and 5 also report on the performance of the CA. We observe that as N or M increases, the CA finds the exact nondominated objective vector set in considerably higher CPU times. The significant increases in the CPU times can be attributed to the increases in the complexity of the integer models that return a single nondominated objective vector and increases in the number of nondominated objective vectors.

The differences between the performances of the BAB algorithm and CA increase as N increases. Note from Table 4 that when N = 300 and M = 600, the average CPU times per nondominated objective vector are 2.593 and 1.01 seconds for the CA and BAB algorithm, respectively. As another notable example, for (N, M) values of (500, 1200), the average CPU times are 5.183 and 1.854 seconds for the CA and BAB algorithm, respectively.

We also observe that the BAB algorithm behaves more consistent than the CA. Note from Table 5 that, for (N, M) values of (500, 1200), the average and maximum CPU times by the BAB algorithm are 306.508 and 561.688 seconds, respectively. For the CA, the respective average and maximum CPU times are 833.533 and 1506.47 seconds.

7. CONCLUSIONS

In this study, we consider a bi-objective CPP with two additive objectives, like total cost and total distance, total distance and total priority, total cost and total time. We propose two algorithms to generate the exact set of all nondominated objective vectors. The Classical Approach uses the optimal solutions of the mixed integer programs and BAB Algorithm uses the optimal solutions of the linear programming relaxations. The BAB Algorithm returns the set of all nondominated objective vectors for problem instances with up to 500 nodes and 1300 arcs in less than one hour and is superior to the Classical Approach.

To the best of our knowledge, our study is the first attempt for the exact solutions of the bi-objective arc routing problems. Our results derived for the CPP can be extended to more general arc routing problems with more than one postman.

REFERENCES

1. **Amine, K. & Djellab, R.** 2013. Industrial and Urban Applications of Eulerian and Chinese Walks, In Graph Theory for Operations Research and Management: Applications in Industrial Engineering, IGI-Global, Hershey,. 271-279.
2. **Brucker, P.** 1981. "The Chinese Postman Problem For Mixed Graphs. Graph Theoretic Concepts in Computer Science," Springer Berlin Heidelberg.
3. **Christofides, N., Benavent, E., Campos, V., Corberán, A., & Mota, E.** 1984. "An Optimal Method for the Mixed Postman Problem." System Modelling and Optimization. Springer Berlin Heidelberg.
4. **Corberán, A., Oswald, M., Plana, I., Reinelt, G., & Sanchis, J. M.** 2012. "New Results on the Windy Postman Problem." Mathematical Programming, vol. 132, p. 309-332.
5. **Dewil, R., Vansteenwegen, P., & Cattrysse, D.** 2011. Cutting path optimization using Tabusearch. Key Engineering Materials, vol. 473, p. 739–748.
6. **Edmonds, J.** 1963. Chinese postmen problem. Operations Research, 13, B73-B77.
7. **Edmonds, J., & Johnson, E. L.** 1973. "Matching, Euler Tour and the Chinese Postman Problem." Mathematical Programming, vol. 5, p. 88-124.
8. **Eiselt, H. A., Gendreau, M., & Laporte, G.** 1995. "Arc

Routing Problems, Part I: The Chinese Postman Problem." Operations Research, vol. 43, p. 231-242.

9. **Ford, L.R.,& Fulkerson,D. R.** 1962. "Flows in Networks," Princeton University Press: Princeton.
10. **Grandinetti, L., Guerriero, F., Laganà, D., & Pisacane, O.** 2012. "An Optimization-Based Heuristic for the Multi-objective Undirected Capacitated Arc Routing Problem." Computers & Operations Research, 39, 2300-2309.
11. **Grötschel, M., & Win, Z.** 1992. "A Cutting Plane Algorithm for the Windy Postman Problem." Mathematical Programming, vol. 55, p. 339-358.
12. **Guan, M.** 1962. Graphic programming using odd or even points. Chinese Math, vol. 110, p. 273-277.
13. **Haimes, Y. Y., Lasdon, L. S., & Wismer, D. A.** 1971. "On a Bicriterion Formulation of the Problems of Integrated System Identification and System Optimization." IEEE Transactions on Systems Man and Cybernetics, vol. 1, p. 296-297.
14. **Han, G., & Na, S.** 1999. "A Study on Torch Path Planning in Laser Cutting Processes Part 2: Cutting Path Optimization Using Simulated Annealing." Journal of Manufacturing Processes, 1, 62–70.
15. **Imahori, S., Kushiya, M., Nakashima, T., & Sugihara, K.** 2008. "Generation of Cutter Paths" Processing Technology, vol. 206 (1-3), p. 453–461.
16. **Lacomme, P., Prins, C., & Sevaux, M.** 2006. "A Genetic Algorithm for a Bi-objective Capacitated Arc Routing Problem." Computers & Operations Research, vol. 33, p. 3473-3493.
17. **Malandraki, C., & Daskin, M. S**. 1993. "The Maximum Benefit Chinese Postman Problem and the Maximum Benefit Traveling Salesman Problem." European Journal of Operational Research, vol. 65, p. 218-234.
18. **Manber, U., & Israni, S.** 1984. "Pierce Point Minimization and Optimal Torch Path Determination in Flame Cutting." Journal of Manufacturing Systems, vol. 3, p.81–89.
19. **Martello, S., & Toth, P.** 1990. "An Exact Algorithm for Large Unbounded Knapsack Problems." Operations Research Letters, vol. 9, p. 15-20.
20. **Mei, Y., Tang, K., & Yao, X.** 2011. "Decomposition-Based Memetic Algorithm for Multiobjective Capacitated Arc Routing Problem." IEEE Transactions on Evolutionary Computation, vo. 15, p. 151-165.
21. **Minieka, E.** 1979. "The Chinese Postman Problem For Mixed Networks." Management Science, vol. 25, p. 643-648.
22. **Nobert, Y. & Picard, J.** 1991. "An Optimal Algorithm for the Mixed Chinese Postman Problem." Centre de Recherche Sur Les Transports Publication.
23. **Papadimitriou, C. H.** 1976. "On the Complexity of Edge Traversing." Journal of the ACM, vol. 23, p. 544-554.
24. **Prakash, S., Sharma, M. K., & Singh, A.** 2009. "A Heuristic For Multi-Objective Chinese Postman Problem." Computers & Industrial Engineering Conference Proceedings, p. 596-599.
25. **Rodrigues, A. M., & Ferreira, J. S.** 2012. "Cutting path as a Rural Postman Problem: Solutions by memetic algorithms." International Journal of Combinatorial Optimization Problems and Informatics, vol. 3, p. 31–46.
26. **Win, Z.** 1989. "On the Windy Postman Problem on Eulerian Graphs." Mathematical Programming, vol. 44, p.97-112.