THE UNIVERSITY OF TEXAS AT SAN ANTONIO, COLLEGE OF BUSINESS

Working Paper SERIES

WP # 0007MSS-061-2010

Date March 2, 2010

A Branch-and-Bound Algorithm for Representative Integer Efficient Solutions in Multiple Objective Network Programming Problems

Minghe Sun

Department of Management Science and Statistics

College of Business

The University of Texas at San Antonio

UTSA COLLEGE OF BUSINESS KNOWLEDGE FOR A NEW WORLD

ONE UTSA CIRCLE
SAN ANTONIO, TEXAS 78249-0631
210 458-4317 | BUSINESS.UTSA.EDU

A Branch-and-Bound Algorithm for Representative Integer Efficient Solutions in Multiple Objective Network Programming Problems

Minghe Sun

Department of Management Science and Statistics
College of Business
The University of Texas at San Antonio
San Antonio, TX 78249-0632
Phone: (210) 458-5777
Fax: (210) 458-6350
minghe.sun@utsa.edu
http://faculty.business.utsa.edu/msun

A Branch-and-Bound Algorithm for Representative Integer Efficient Solutions in Multiple Objective Network Programming Problems

Abstract

In many applications of multiple objective network programming problems, only integer solutions are acceptable as the final optimal solution. Representative efficient solutions are usually obtained by sampling the efficient set through the solution of augmented weighted Tchebycheff network programs. Because such efficient solutions are usually not integer solutions, a branch-and-bound algorithm is developed to find integer efficient solutions. The purpose of the branch-and-bound algorithm is to support interactive procedures by generating representative integer efficient solutions. To be computationally efficient, the algorithm takes advantage of the network structure as much as possible. An algorithm, used in the branch-and-bound algorithm and performed on the spanning tree, is developed to construct feasible solutions from infeasible solutions and basic solutions from non-basic solutions when bounds on branching variables change. The branch-and-bound algorithm finds either supported or unsupported integer efficient solutions as long as they are optimal. Details of the algorithm are presented, an example is provided and computational results are reported. Computational results show that the algorithm is very powerful.

Keywords: **Network Flow Algorithms; Multiple Objective Programming; Multiple Criteria Decision Making; Integer Programming; Branch-and-Bound**

JEL Code: **C14, C61**

A Branch-and-Bound Algorithm for Representative Integer Efficient Solutions in Multiple Objective Network Programming Problems

1. Introduction

Single objective network programming (NP) models have been studied extensively and have been applied to many real life problems [Ahuja, Magnanti and Orlin, 1993; Glover, Klingman and Phillips, 1992; Kennington and Helgason, 1980; Murty, 1992]. Computationally very efficient algorithms have been developed for single objective NP models. Multiple objective network programming (MONP) problems also attracted the attention of researchers and practitioners because many applications have multiple objectives in nature [Hamacher, Pedersen and Ruzika, 2007]. Current and Min [1986], Current and Marsh [1993], Ehrgott and Gandibleux [2000] and Hamacher, Pedersen and Ruzika [2007] provided comprehensive reviews of MONP studies. In many applications, only integer solutions are acceptable. With the integrality requirement, MONP problems become multiple objective integer network programming (MOINP) problems. MOINP problems are much more difficult to solve than single objective NP problems [Steuer and Piercy, 2000; Hamacher, Pedersen and Ruzika, 2007].

Interactive procedures are usually used to solve multiple objective programming (MOP) problems. Many interactive procedures have been developed in the last 40 years [Steuer, 1986]. Based on these earlier pioneering works, many more interactive procedures have been developed more recently. The ideas in some of these interactive procedures can be borrowed to solve MOINP problems if computationally efficient algorithms are available to generate representative integer efficient solutions. Usually representative efficient solutions are sampled and presented to the decision maker (DM) for evaluation in an interactive procedure [Steuer and Choo, 1983; Steuer, Silverman and Whisman, 1993; Sun, Stam and Steuer, 1996, 2000]. The preference information elicited from the DM through the evaluation of representative efficient solutions is then incorporated into the solution process in order to search for better solutions. More representative efficient solutions are then sampled and presented to the DM for evaluation. This process continues until some preset criteria are met or until the DM is satisfied with a solution. In an interactive procedure, most of the computational effort is used to generate representative efficient solutions.

For MOINP problems, representative integer efficient solutions can be generated by solving augmented weighted Tchebycheff integer network programs (AWTINPs), subproblems derived from the MOINP problem [Drinka, Sun and Murry 1996; Steuer, 1986; Sun, 2003, 2005a, 2009]. Dropping the integrality requirement, an AWTINP is relaxed to an augmented weighted Tchebycheff network program (AWTNP). In this study, a branch-and-bound (BB) algorithm is developed to solve AWTINPs so as to generate representative integer efficient solutions for MOINP problems. In the BB algorithm, an AWTINP is solved by solving a series of AWTNPs. Integer efficient solutions in MOINP problems may be basic and non-basic or may be supported or unsupported. The BB algorithm is capable of finding both basic and non-basic as well as supported and unsupported integer efficient solutions. However, the purpose of the BB algorithm is not to completely enumerate all integer efficient solutions in an MOINP problem but is to generate representative integer efficient solutions to support interactive procedures.

The BB algorithm is a general solution approach for various optimization problems and is the standard solution approach for integer programming problems. While the concept of the BB algorithm is not complicated, the major contribution of this study is to make the BB algorithm computationally efficient by using the network structure of the MOINP problem and by developing algorithms to restore feasibility and to transform non-basic solutions to basic solutions when subproblems are constructed and after subproblems are solved. Computational results show that the BB algorithm can solve pretty large MOINP problems.

The generation of integer efficient solutions for MOINP problems has attracted the attention of researchers. Hamacher, Pedersen and Ruzika [2007] provided a review of the studies in this area. Most of the studies focused on biobjective MOINP problems [*e.g.*, Pulat, Huarng, and Lee, 1992; Lee and Pulat, 1993; Sedeño-Noda and González-Martín, 2000, 2001, 2003; Przybylski, Gandibleux and Ehrgott, 2006; Eusébio and Figueira, 2009b; and Raith and Ehrgott, 2009]. The purpose of these studies is to find all integer efficient solutions for such problems although some earlier studies can only find basic and some others can only find supported integer efficient solutions. Later studies [Przybylski, Gandibleux and Ehrgott, 2006; Eusébio and Figueira, 2009b; and Raith and Ehrgott, 2009] also provided reviews and examined the flaws of other earlier studies. Özlen and Azizoğlu [2009] developed a general approach to generate all integer efficient solutions for integer linear MOP problems. This general approach is applicable to MOINP problems. Eusébio and Figueira [2009a] developed a method to enumerate all supported integer efficient solutions of MOINP problems.

These methods make the enumeration of all integer efficient solutions theoretically possible although they may not be practically feasible [Eusébio and Figueira, 2009a]. Biobjective MOINP problems may have an exponential number of basic efficient solutions [Ruhe, 1998; Hamacher, Pedersen and Ruzika, 2007]. A general MOINP problem with more objective functions may be much more complicated. Because of the huge number of integer efficient solutions in any reasonable application, finding all integer efficient solutions is an unmanageable task. Even though all the integer efficient solutions are found, managing them is another unmanageable task. After the whole set of integer efficient solutions has been found, sophisticated methods, possibly interactive, may be needed to find a single final solution. In practical applications, complete enumeration of all integer efficient solutions may be unnecessary because only one optimal solution is needed for an MOINP problem.

Methods for the generation of efficient basic solutions for linear MOP problems, such as ADBASE [Steuer, 2003], EFFACET [Isermann, 1997, 1984] and the approach recently developed by Przybylski, Gandibleux and Ehrgott [2009], may be used to generate basic integer efficient solutions for MOINP problems. However, basic integer efficient solutions are only a small portion of the whole set of integer efficient solutions. Although basic solutions of NP problems are integer solutions, the optimal solution of an MOINP problem is not necessarily a basic solution. Searching only basic solutions may miss the optimal solution of an MOINP problem.

Procedures are also developed to construct integer efficient solutions from fractional efficient solutions for MOINP problems. Mustafa and Goh [1998] proposed such an approach for MOINP problems with two or three objective functions from fractional efficient solutions generated with DINAS [Ogryczak, Studzi·ski and Zorychta, 1989, 1992]. This approach works for problems with two objective functions, but is problematic for problems with three objective functions, and is impossible to generalize to problems with more than three objective functions. Sun

[2009] developed an enumerative algorithm to find representative integer efficient solutions that approximately solve the AWTINP. After finding the optimal solution of an AWTNP, this algorithm searches the neighborhood to find the best supported integer efficient solution. The CPU time taken by this algorithm is hardly noticeable but it searches only supported integer efficient solutions. For practical purpose, these supported integer efficient solutions are sufficient but theoretically unsupported integer efficient solutions also need to be searched. As shown by Sedeño-Noda and González-Martín [2001] and Raith and Ehrgott [2009] through computational experiments for biobjective MOINP problems, the number of unsupported integer efficient solutions is usually much larger than that of supported integer efficient solutions. In general multiple objective combinatorial problems, supported efficient solutions are often only a small portion of the complete set of efficient solutions even for the biobjective case [Przybylski, Gandibleux and Ehrgott, 2009].

2. The Multiple Objective Integer Network Programming Problem

A network is a set of nodes N connected with a set of directed arcs A. The number of nodes is denoted by $n = |\mathrm{N}|$ and the number of arcs is denoted by $a = |\mathrm{A}|$. An arc in the network directed from node $i \in \mathrm{N}$ to node $j \in \mathrm{N}$ is denoted by $(i, j) \in \mathrm{A}$. Nodes usually represent locations in physical networks such as factories, warehouses, distribution centers and customers in a distribution network, or servers, gateways and terminals in a computer network. Arcs usually represent connections such as a road between two locations in a distribution network or communication channels in a computer network.

The flow on (i, j) is represented by x_{ij} and the lower and upper bounds on x_{ij} are represented by l_{ij} and u_{ij}, respectively. The default $l_{ij} = 0$ for all $(i, j) \in \mathrm{A}$ is used in the implementation of NP algorithms in this study. Each x_{ij} is required to be an integer in a feasible solution of an MOINP problem. The net requirement for the goods to be shipped at node i is represented by b_i. Node i is a supply node if $b_i > 0$, a demand node if $b_i < 0$, or a transshipment node if $b_i = 0$. The b_i for each $i \in \mathrm{N}$ and the l_{ij} and the u_{ij} for each $(i, j) \in \mathrm{A}$ are all assumed to be integers. The number of objective functions in the MOINP model is represented by K. Each objective function represents one type of cost and all objective functions are to be minimized. The cost of type k of shipping one unit of goods along (i, j) is represented by c_{ij}^k. The vector of unit costs of the K objective functions for (i, j) is represented by $\mathbf{c}_{ij} = (c_{ij}^1, c_{ij}^2, \cdots, c_{ij}^K)$. The x_{ij} s are the variables, called flow variables, and all others are known parameters in the MOINP model. An MOINP model can be stated formally as

$$min \quad \sum_{(i,j)\in \mathrm{A}} c_{ij}^k x_{ij}, \qquad \text{for } k = 1, ..., K \tag{1}$$

$$s.t. \quad \sum_{\{j|(i,j)\in \mathrm{A}\}} x_{ij} - \sum_{\{j|(j,i)\in \mathrm{A}\}} x_{ji} = b_i, \qquad \text{for } i \in \mathrm{N} \tag{2}$$

$$l_{ij} \le x_{ij} \le u_{ij}, \qquad \text{for } (i, j) \in \mathrm{A} \tag{3}$$

$$x_{ij} \text{ integers}, \qquad \text{for } (i, j) \in \mathrm{A} \tag{4}$$

Although all the objective functions (1) are to be minimized simultaneously, not all of them can achieve their minima at the same time in a usual MOINP problem. The conservation of flow constraints (2) and the bound constraints (3) are the same as those in a single objective NP problem. If fractional solutions are acceptable in an application, the integrality constraints (4) are dropped from the model. Without these integrality constraints, the model in (1)-(3) is a regular MONP model and is a relaxation of the corresponding MOINP model.

Using vector and matrix notation, the vector of flow variables is denoted by $\mathbf{x} \in \Re^a$, *i.e.*, $\mathbf{x} = \{x_{ij} \mid (i, j) \in \mathrm{A}\}$; the lower and upper bound vectors of $\mathbf{x}$ are denoted by $\mathbf{l} \in \Re^a$ and $\mathbf{u} \in \Re^a$, respectively; the cost coefficient matrix is denoted by $C \in \Re^{K \times a}$; the node-arc incidence matrix is denoted by $A \in \Re^{n \times a}$; and the node requirement vector is denoted by $\mathbf{b} \in \Re^n$. The MONP model can be written as $min\{C\mathbf{x} \mid \mathbf{x} \in X\}$ with $X = \{\mathbf{x} \in \Re^a \mid A\mathbf{x} = \mathbf{b} \text{ and } \mathbf{l} \le \mathbf{x} \le \mathbf{u}\}$. A $\mathbf{x} \in X$ is a feasible solution and $X \subset \Re^a$ is the feasible region of the MONP problem in decision space. Similarly, the MOINP model can be written as $min\{C\mathbf{x} \mid \mathbf{x} \in X^I\}$ with $X^I = \{\mathbf{x} \in \Re^a \mid A\mathbf{x} = \mathbf{b}, \mathbf{l} \le \mathbf{x} \le \mathbf{u} \text{ and } \mathbf{x} \text{ integers}\}$. A $\mathbf{x} \in X^I$ is a feasible solution and $X^I \subset \Re^a$ is the feasible region of the MOINP problem in decision space. Apparently, X^I is a subset of X and X is the convex hull of X^I. Any $\mathbf{x}$, such that $\mathbf{x} \in X$ but $\mathbf{x} \notin X^I$, is a fractional or non-integer solution.

A $\mathbf{z} \in \Re^K$, such that $\mathbf{z} = C\mathbf{x}$, is a criterion vector. The set of all feasible criterion vectors $Z \subset \Re^K$ ($Z^I \subset \Re^K$), such that $Z = \{\mathbf{z} \in \Re^K \mid \mathbf{z} = C\mathbf{x} \text{ and } \mathbf{x} \in X\}$ ($Z^I = \{\mathbf{z} \in \Re^K \mid \mathbf{z} = C\mathbf{x} \text{ and } \mathbf{x} \in X^I\}$), is the feasible region and a $\mathbf{z} \in Z$ ($\mathbf{z} \in Z^I$) is a feasible solution of the MONP (MOINP) problem in criterion space. In an MOINP problem, the components of a $\mathbf{z} \in Z^I$ are not necessarily integers although those of a $\mathbf{x} \in X^I$ are.

A $\bar{\mathbf{z}} \in Z$ ($\bar{\mathbf{z}} \in Z^I$) is a nondominated criterion vector if there does not exist any $\mathbf{z} \in Z$ ($\mathbf{z} \in Z^I$) such that $\mathbf{z} \le \bar{\mathbf{z}}$ and $\mathbf{z} \ne \bar{\mathbf{z}}$ [Steuer, 1986]. A $\mathbf{z} \in Z$ ($\mathbf{z} \in Z^I$) is dominated if it is not nondominated. The set of all nondominated criterion vectors of the MONP (MOINP) problem is represented by Z (Z^I). A $\bar{\mathbf{x}} \in X$ ($\bar{\mathbf{x}} \in X^I$) is an efficient solution if its criterion vector $\bar{\mathbf{z}} = C\bar{\mathbf{x}}$ is nondominated. A $\mathbf{x} \in X$ ($\mathbf{x} \in X^I$) is inefficient if it is not efficient. The set of all efficient solutions of the MONP (MOINP) problem is represented by X (X^I). Apparently $\mathrm{Z}^I \ne \mathrm{Z}$ and $\mathrm{X}^I \ne \mathrm{X}$. A $\mathbf{x} \in \mathrm{X}$ is not in X^I if it does not satisfy the integrality constraints (4) and a $\mathbf{x} \in \mathrm{X}^I$ is not in X if $\mathbf{z} = C\mathbf{x}$ is dominated by the criterion vectors of some fractional solutions.

The set $Z^{\geq}$ is defined as $Z^{\geq} = \text{convex hull of } \{\mathrm{Z}^I \oplus \{\mathbf{z} \in \Re^K \mid \mathbf{z} \ge 0\}\}$ [Steuer, 1986]. If defined on Z, $Z^{\geq}$ is the same. A $\mathbf{z} \in \mathrm{Z}$ ($\mathbf{z} \in \mathrm{Z}^I$) is a supported nondominated criterion vector if it is on the boundary of $Z^{\geq}$ and is an unsupported nondominated criterion vector otherwise [Steuer, 1986]. A $\mathbf{x} \in \mathrm{X}$ ($\mathbf{x} \in \mathrm{X}^I$) is a supported (unsupported) efficient solution if $\mathbf{z} = C\mathbf{x}$ is supported (unsupported). All $\mathbf{z} \in \mathrm{Z}$ are supported although some $\mathbf{z} \in \mathrm{Z}^I$ are unsupported. A $\mathbf{x}$ ($\mathbf{z}$), such that $\mathbf{x} \in \mathrm{X}^I$ ($\mathbf{z} \in \mathrm{Z}^I$) but $\mathbf{x} \notin \mathrm{X}$ ($\mathbf{z} \notin \mathrm{Z}$), is an unsupported efficient solution (nondominated criterion vector) of the MOINP problem. An unsupported nondominated criterion vector of the MOINP problem is dominated by the criterion vectors of some fractional solutions of the MONP problem. All

unsupported integer efficient solutions are non-basic or basic but their criterion vectors are dominated by those of some fractional solutions.

The ideal point $\mathbf{z}^*$ is defined by $z_k^* = min\{z_k \mid \mathbf{z} \in Z\}$. Usually $\mathbf{z}^* \notin Z$, *i.e.*, there is not a feasible solution, fractional or integer, that minimizes all objective functions simultaneously [Steuer, 1986; Sun, 2005b]. A utopian point $\mathbf{z}^{**}$, defined on $\mathbf{z}^*$, *i.e.*, $z_k^{**} = z_k^* - \varepsilon_k$ with $\varepsilon_k > 0$ and small for all k [Steuer, 1986], is usually used as a reference in the solution process of an MONP problem. A $\hat{\mathbf{x}} \in X^I$, such that $\hat{\mathbf{z}} = C\hat{\mathrm{x}}$ is most preferred among all $\mathbf{z} \in Z^I$ by the DM, is an optimal solution of the MOINP problem in (1)-(4). Theoretically, an optimal solution maximizes the DM's value function if such a value function exists [Steuer, 1986]. Because an optimal solution must be efficient [Steuer, 1986], only those $\mathbf{x}$, such that $\mathbf{x} \in \mathrm{X}^I$, are candidates for an optimal solution.

The integrality constraints (4) are implicitly implied in single objective NP problems. Because of the total unimodularity property of A [Ahuja, Magnanti and Orlin, 1993; Bazaraa, Jarvis and Sherali, 1990; Nemhauser and Wolsey, 1988], all basic solutions of the MONP problem are integer solutions if b_i for all $i \in \mathrm{N}$ and l_{ij} and u_{ij} for all $(i, j) \in \mathrm{A}$ are integers. Therefore, all basic solutions of the MONP problem are feasible solutions of the MOINP problem. However, these integrality constraints must be enforced for MOINP problems because not every efficient solution of the MONP problem is basic with integer values. If the DM's value function is nonlinear, optimal solutions are not necessarily basic and, therefore, non-basic efficient solutions need to be examined.

3. Generation of Integer Efficient Solutions

In this section, the AWTINP (AWTNP) is discussed. Subproblems obtained through the separation of an AWTNP, also AWTNPs, are described. Finally, basis exchanges used in the BB algorithm are briefly explained.

3.1. The Augmented Weighted Tchebycheff Integer Network Program

The weighting vector space Λ is defined as $\Lambda = \{\boldsymbol{\lambda} \in \Re^K \mid \lambda_k > 0, \text{ for all } k = 1,...,K, \text{ and } \sum_{k=1}^K \lambda_k = 1\}$ [Steuer, 1986]. Given a $\boldsymbol{\lambda} \in \Lambda$, an AWTINP formulated from the MOINP model in (1)-(4) is

$$min \quad \alpha + \rho \sum_{k=1}^{K} \left(\sum_{(i,j)\in \mathrm{A}} c_{ij}^k x_{ij} - z_k^{**} \right) \tag{5}$$

$$s.t. \quad \sum_{\{j \mid (i,j)\in \mathrm{A}\}} x_{ij} - \sum_{\{j \mid (j,i)\in \mathrm{A}\}} x_{ji} = b_i, \qquad \text{for } i \in \mathrm{N} \tag{6}$$

$$\sum_{(i,j)\in \mathrm{A}} c_{ij}^k x_{ij} - \frac{1}{\lambda_k}\alpha + s_k = z_k^{**}, \qquad \text{for } k = 1,...,K \tag{7}$$

$$l_{ij} \le x_{ij} \le u_{ij}, \qquad \text{for } (i, j) \in \mathrm{A} \tag{8}$$

$$s_k \ge 0, \qquad \text{for } k = 1,...,K \tag{9}$$

$$x_{ij} \text{ integers}, \qquad \text{for } (i, j) \in \mathrm{A} \tag{10}$$

$$\alpha \text{ unrestricted.} \tag{11}$$

Dropping the integrality constraints (10), the AWTINP is relaxed to an AWTNP. In the AWTINP (AWTNP), $\rho > 0$ is such a small scalar that the second term in (5) is significant but not dominating, the s_k s, for $k = 1,...,K$, are the slack variables, and α is the $\boldsymbol{\lambda}-$weighted Tchebycheff metric between $\mathbf{z}^{**}$ and a $\mathbf{z} \in Z$. Because $\mathbf{z} \in Z$ is not in the model, it can be maintained and updated in the solution process or can be evaluated using (1) when needed. The value of α for any given $\mathbf{z} \in Z$ and any given $\boldsymbol{\lambda} \in \Lambda$ can be computed using $\alpha = \| \mathbf{z} - \mathbf{z}^{**} \|_{\lambda}^{\infty} = max\{\lambda_k (z_k - z_k^{**}) \mid k = 1,...,K\}$. With α and $\mathbf{z}$, the objective function (5), called the augmented $\boldsymbol{\lambda}-$weighted Tchebycheff metric, can be evaluated with $d = \| \mathrm{z} - \mathrm{z}^{**} \|_{\lambda}^{\infty} = \alpha + \rho \sum_{k=1}^{K} (z_k - z_k^{**})$.

The K constraints (7) converted from the K objective functions (1) are not network constraints and the $K+1$ variables, *i.e.*, s_k for $k = 1,...,K$ and α, are non-flow variables. Hence, the AWTNP is a network flow problem with side constraints. In the implementation, AWTNPs are solved with the special simplex method for network problems with side constraints [Chen and Saigal, 1977]. Because the integrality constraints (10) are relaxed, an optimal solution of an AWTNP is usually fractional.

A basic solution of an AWTNP has $n-1+K$ basic variables, including both flow and non-flow variables. In the special simplex method for network problems with side constraints, a key tree T is used to partially represent a basic solution of the AWTNP. For easy reference, a node i in T is denoted by $i \in \mathrm{T}$ and an arc (i, j) in T is denoted by $(i, j) \in \mathrm{T}$. Graphically, T grows upward. In the solution process, T is updated iteration after iteration. The $n-1$ arcs in T are key basic arcs and the $n-1$ corresponding flow variables are key basic flow variables. The other K basic variables, called non-key basic variables, may include both flow and non-flow variables. Arcs corresponding to non-key basic flow variables are non-key basic arcs. All other arcs are non-basic arcs. Key basic arcs are further divided into essential and inessential key basic arcs. An essential key basic arc cannot and an inessential key basic arc can be replaced by a non-key basic arc to form a different key tree for the same basic solution. Because α is always a non-key basic variable, the number of non-key basic arcs in any basic solution is no more than $K-1$. The special simplex method for network problems with side constraints searches basic solutions of the AWTNP.

A basic optimal solution of the AWTNP for a given $\boldsymbol{\lambda} \in \Lambda$ is denoted by $(\tilde{\mathbf{x}}_{\lambda}, \tilde{\mathbf{z}}_{\lambda}, \tilde{\mathbf{s}}, \alpha) \in \Re^{a+k+1}$, where $\tilde{\mathbf{x}}_{\lambda} \in \mathrm{X}$, $\tilde{\mathbf{z}}_{\lambda} = C\tilde{\mathbf{x}}_{\lambda} \in \mathrm{Z}$, *i.e.*, $\tilde{\mathbf{x}}_{\lambda}$ is efficient and $\tilde{\mathbf{z}}_{\lambda}$ is nondominated, and $\tilde{\mathbf{s}} = \{\tilde{s}_k \mid k = 1,...,K\}$. The specific $\tilde{\mathbf{x}}_{\lambda}$ and $\tilde{\mathbf{z}}_{\lambda}$ obtained are dependent upon the relative magnitudes of the components of $\boldsymbol{\lambda}$. By using different $\boldsymbol{\lambda} \in \Lambda$, any $\mathbf{x} \in \mathrm{X}$ can be found through an AWTNP. By using a set of widely dispersed $\boldsymbol{\lambda}$ in Λ (or in a reduced subset of Λ), widely dispersed representative efficient solutions can be generated from X (or from a subset of X) [Steuer, 1986]. Although $(\tilde{\mathbf{x}}_{\lambda}, \tilde{\mathbf{z}}_{\lambda}, \tilde{\mathbf{s}}, \alpha)$ is basic in the AWTNP, $\tilde{\mathbf{x}}_{\lambda}$ is not necessarily basic in the MONP problem. The AWTNP is capable of generating basic and non-basic efficient solutions for the MONP problem. However, the non-basic efficient solutions obtained are not necessarily integer solutions. A $\mathbf{x} \in \mathrm{X}^I$ but $\mathbf{x} \notin \mathrm{X}$ cannot be reached by an AWTNP formulated with any $\boldsymbol{\lambda} \in \Lambda$. Hence, a BB algorithm is used to find the optimal solution of the AWTINP.

3.2. Branching and Subproblems

Sun [2009] showed that only the non-key basic flow variables need to be integers for a solution to be integer. Once all non-key basic flow variables become integers and all other variables are adjusted accordingly to preserve feasibility, the resulting solution is an integer solution. As a result, only the integralities of the non-key basic flow variables need to be checked and only non-key basic flow variables are candidates for branching variables.

If (p,q) is non-key basic with a non-integer flow $x_{pq} = \tilde{x}_{pq}$ in an optimal solution of the current AWTNP and x_{pq} is selected as the branching variable, setting x_{pq} to the nearest integers separates the current AWTNP into two subproblems, one with $x_{pq} \le \lfloor \tilde{x}_{pq} \rfloor$ and the other with $x_{pq} \ge \lceil \tilde{x}_{pq} \rceil$. Each subproblem is an AWTNP. Adding $x_{pq} \le \lfloor \tilde{x}_{pq} \rfloor$ is equivalent to resetting u_{pq} to $u_{pq} = \lfloor \tilde{x}_{pq} \rfloor$ and adding $x_{pq} \ge \lceil \tilde{x}_{pq} \rceil$ is equivalent to resetting l_{pq} to $l_{pq} = \lceil \tilde{x}_{pq} \rceil$. Treating these extra constraints as bounds on variables does not increase the size of the resulting AWTNPs. The current AWTNP is the parent problem and the two resulting AWTNPs are the child problems. After separation, some unsupported integer efficient solutions of the parent AWTNP become supported integer efficient solutions of the child AWTNPs and, therefore, can be identified as the optimal solutions of the child AWTNPs. Separation is recursive, *i.e.*, the same flow variable may be selected as the branching variable iteration after iteration before a branch is fathomed.

Because default values of $l_{ij} = 0$ are used in the implementation of network algorithms, a variable substitution is used to handle the bound $x_{pq} \ge \lceil \tilde{x}_{pq} \rceil$. The variable substituting x_{pq} is represented by x'_{pq}, with $x_{pq} = x'_{pq} + \lceil \tilde{x}_{pq} \rceil$. Hence, $0 \le x'_{pq} \le u_{pq} - \lceil \tilde{x}_{pq} \rceil$ when $\lceil \tilde{x}_{pq} \rceil \le x_{pq} \le u_{pq}$. This substitution is equivalent to reducing b_p and increasing b_q by $\lceil \tilde{x}_{pq} \rceil$ units while shipping $\lceil \tilde{x}_{pq} \rceil$ units along (p,q). Substituting x'_{pq} for x_{pq} in the child AWTNP shifts all $\mathbf{z} \in Z$ and $\mathbf{z}^{**}$ to the left by $\mathbf{c}_{pq} \lceil \tilde{x}_{pq} \rceil$. As (12) and (16) in the following show, the actual criterion vector is $\mathbf{z}' = \mathbf{z} - \mathbf{c}_{pq} \lceil \tilde{x}_{pq} \rceil$ and the actual utopian point is $\mathbf{z}'^{**} = \mathbf{z}^{**} - \mathbf{c}_{pq} \lceil \tilde{x}_{pq} \rceil$ in the resulting child AWTNP. After x_{pq} is replaced by $x'_{pq} + \lceil \tilde{x}_{pq} \rceil$, the AWTNP in (12)-(20) in the following is obtained

$$\textit{min} \quad \alpha + \rho \sum_{k=1}^{K} \left(\sum_{(i,j) \in \mathrm{A} \setminus (p,q) \setminus} c_{ij}^k x_{ij} + c_{pq}^k x'_{pq} - \left(z_k^{**} - c_{pq}^k \lceil \tilde{x}_{pq} \rceil \right) \right) \tag{12}$$

$$\textit{s.t.} \quad \sum_{\{j \mid (i,j) \in \mathrm{A}\}} x_{ij} - \sum_{\{j \mid (j,i) \in \mathrm{A}\}} x_{ji} = b_i \qquad \text{for } i \in \mathrm{N} \setminus \{p,q\} \tag{13}$$

$$\sum_{\{j \mid (p,j) \in \mathrm{A} \setminus (p,q)\}} x_{pj} + x'_{pq} - \sum_{\{j \mid (j,p) \in \mathrm{A}\}} x_{jp} = b_p - \lceil \tilde{x}_{pq} \rceil \tag{14}$$

$$\sum_{\{j \mid (q,j) \in \mathrm{A}\}} x_{qj} - \sum_{\{j \mid (j,q) \in \mathrm{A}\} \setminus (p,q)\}} x_{jq} - x'_{pq} = b_q + \lceil \tilde{x}_{pq} \rceil \tag{15}$$

$$\sum_{(i,j) \in \mathrm{A} \setminus (p,q)} c_{ij}^k x_{ij} + c_{pq}^k x'_{pq} - \frac{1}{\lambda_k} \alpha + s_k = z_k^{**} - c_{pq}^k \lceil \tilde{x}_{pq} \rceil \qquad \text{for } k = 1, \ldots, K \tag{16}$$

$$l_{ij} \le x_{ij} \le u_{ij} \quad \text{for } (i,j) \in \mathrm{A} \setminus (p,q) \tag{17}$$

$$0 \le x'_{pq} \le u_{ij} - \lceil \tilde{x}_{pq} \rceil \tag{18}$$

$$s_k \ge 0 \quad \text{for } k = 1,\ldots,K \tag{19}$$

$$\alpha \text{ unrestricted} \tag{20}$$

3.3. Basis Exchange

Although a series of AWTNPs is solved to solve an AWTINP, only one AWTNP is stored in memory and is solved at a time. The differences in these AWTNPs are in the bounds on the branching variables. Each AWTNP is constructed from the previous one by changing the bounds on the branching variables and is solved from the optimal solution of the previous one. After the bound of a branching variable changes, the optimal solution of the previous AWTNP is not feasible in the current AWTNP. After restoring the original bounds of a branching variable, the resulting solution may be non-basic in the current AWTNP. Basis exchanges are used to increase or decrease the flows on branching variables in order to find a feasible solution from an infeasible solution or to find a basic solution from a non-basic solution.

A basis exchange is similar to one iteration in the special simplex method for network problems with side constraints. An entering arc is brought into T to replace an arc already in T. The entering arc may be non-basic or non-key basic. If the entering arc is a non-basic arc $(p,q) \notin \mathrm{T}$, its flow can only increase if $x_{pq} = l_{pq}$ or can only decrease if $x_{pq} = u_{pq}$. If the entering arc is a non-key basic arc $(p,q) \notin \mathrm{T}$, its flow may increase or decrease if $l_{pq} < x_{pq} < u_{pq}$. For the entering arc $(p,q) \notin \mathrm{T}$, there is a unique path in T. This unique path and (p,q) form a fundamental circle. After a leaving arc on the fundamental circle is determined, T and the non-key basic variables are updated in the basis exchange.

4. Algorithms

Although conceptually the BB algorithm is pretty simple, the feasibility and basis restorations are somewhat complicated when the bounds on branching variables change. In this section, feasibility and basis restorations are discussed before the BB algorithm is described.

4.1. Feasibility and Basis Restorations

When x_{pq} is selected as the branching variable, the subproblem with $x_{pq} \le \lfloor \tilde{x}_{pq} \rfloor$ is solved first. With this new upper bound on x_{pq}, the optimal solution of the parent problem with $x_{pq} = \tilde{x}_{pq}$ is infeasible in this subproblem. After adjusting the flows by the amount $\tilde{x}_{pq} - \lfloor \tilde{x}_{pq} \rfloor$ on all arcs on the fundamental circle formed by (p,q) and adjusting the non-key basic variables accordingly, the solution obtained is feasible but not necessarily basic in the AWTNP. Therefore, a basis exchange is used to obtain a basic solution satisfying $x_{pq} \le \lfloor \tilde{x}_{pq} \rfloor$. Reoptimization then starts from this feasible basic solution. In the basis exchange, (p,q) is the entering arc and the

flow on (p,q) is reduced. If the branch cannot be fathomed after the AWTNP is solved, branching will continue by selecting a non-integer non-key basic flow variable as the new branching variable.

The subproblem with $x_{pq} \geq \lceil \tilde{x}_{pq} \rceil$ cannot be constructed and solved from the optimal solution of the parent problem. After the branch with $x_{pq} \leq \lfloor \tilde{x}_{pq} \rfloor$ has been fathomed and before the branch with $x_{pq} \geq \lceil \tilde{x}_{pq} \rceil$ is examined, x_{pq} may be in any status in the current solution, *i.e.*, at its lower bound with $x_{pq} = l_{pq}$, at its current upper bound with $x_{pq} \leq \lfloor \tilde{x}_{pq} \rfloor$, or key or non-key basic with $l_{pq} \leq x_{pq} \leq \lfloor \tilde{x}_{pq} \rfloor$. In any case, the current solution is not feasible in the subproblem with $x_{pq} \geq \lceil \tilde{x}_{pq} \rceil$ because it satisfies $x_{pq} \leq \lfloor \tilde{x}_{pq} \rfloor$. Hence, the flow on (p,q) is increased to satisfy $x_{pq} \geq \lceil \tilde{x}_{pq} \rceil$ in basis exchanges. The original upper bound u_{pq} is restored first before basis exchanges are made and before the AWTNP in (12)-(20) is constructed. If x_{pq} is non-basic or is non-key basic, (p,q) is the entering arc and x_{pq} increases in a basis exchange. However, if x_{pq} is key basic, the basis exchange is much more involved. A non-key basic or a non-basic arc has to be selected as the entering arc to increase x_{pq}.

If $(p,q) \in \mathrm{T}$ is removed from T, T is divided into two subtrees denoted by T_p and T_q, respectively. An algorithm is developed to find an arc $(s,t) \notin \mathrm{T}$ with one end in T_p and the other in T_q as the entering arc to increase x_{pq}. If $s \in \mathrm{T}_p$ and $t \in \mathrm{T}_q$, x_{pq} increases when (s,t) is brought into T and x_{st} decreases. However, x_{st} can decrease only if $x_{st} > 0$, *i.e.*, when (s,t) is non-key basic or is non-basic with $x_{st} = u_{st}$. If $s \in \mathrm{T}_q$ and $t \in \mathrm{T}_p$, x_{pq} increases when (s,t) is brought into T and x_{st} increases. However, x_{st} can increase only if $x_{st} < u_{st}$, *i.e.*, when (s,t) is non-key basic or is non-basic with $x_{st} = l_{st}$. The algorithm in the following searches for such an entering arc. It searches the non-key basic arcs before searching the non-basic arcs. The following is a step-by-step description of this algorithm.

Step 1. If $(p,q) \in \mathrm{T}$ points downward, let $\mu_i = 1$ for each $i \in \mathrm{T}_p$ and $\mu_i = 0$ for each $i \notin \mathrm{T}_p$; otherwise, let $\mu_i = -1$ for each $i \in \mathrm{T}_q$ and $\mu_i = 0$ for each $i \notin \mathrm{T}_q$. Let $k = 1$.

Step 2. If $k > K$, let $k = 1$ and go to Step 5; otherwise, go to Step 3.

Step 3. If the k th non-key basic variable is not a flow variable, let $k = k + 1$ and go to Step 2; otherwise, go to Step 4.

Step 4. Suppose the k th non-key basic variable is the flow variable x_{st}. If $\mu_s - \mu_t < 0$ and $x_{st} < u_{st}$ or if $\mu_s - \mu_t > 0$ and $x_{st} > 0$, go to Step 8; otherwise, let $k = k + 1$ and go to Step 2.

Step 5. If $k > m$, Stop; otherwise, go to Step 6.

Step 6. If the k th arc is basic, let $k = k + 1$ and go to Step 5; otherwise, go to Step 7.

Step 7. Suppose the k th arc is the non-basic arc (s,t). If $\mu_s - \mu_t < 0$ and $x_{st} = l_{st}$ or if $\mu_s - \mu_t > 0$ and $x_{st} = u_{st}$, go to Step 8; otherwise, let $k = k + 1$ and go to Step 5.

Step 8. Make a basis exchange. Let (s,t) be the entering arc, and let the flow on (s,t) increase if $\mu_s - \mu_t < 0$ or decrease if $\mu_s - \mu_t > 0$ in the basis exchange. Stop.

One basis exchange is necessary but more basis exchanges are needed if $x_{pq} \geq \lceil \tilde{x}_{pq} \rceil$ is not satisfied after the first basis exchange. The algorithm above is used repeatedly until $x_{pq} \geq \lceil \tilde{x}_{pq} \rceil$ is satisfied. If the branch with $x_{pq} \geq \lceil \tilde{x}_{pq} \rceil$ cannot be fathomed after the AWTNP is solved, branching will continue by selecting a non-integer non-key basic flow variable as the new branching variable.

4.2. The Branch-and-Bound Algorithm

The best supported integer efficient solution found with the enumerative algorithm of Sun [2009] is used as the initial incumbent solution. Without this initial incumbent solution, the behavior of the BB algorithm is not predictable. It may branch on several flow variables until reaching their original lower or upper bounds before finding the first integer solution. The optimal solution obtained by the BB algorithm for the AWTINP in (5)-(11) is at least as good as this initial incumbent solution.

In other types of integer programming problems, a branch can be fathomed in three ways, *i.e.*, the subproblem does not have a feasible solution, the optimal solution of the subproblem is integer, or the value of the objective function of the optimal solution is worse than that of the incumbent solution [Nemhauser and Wolsey, 1988]. In an AWTINP, a subproblem formed by branching on a non-integer non-key basic flow variable always has feasible solutions. Hence, a branch can be fathomed in only two ways. If a branch is fathomed because the optimal solution of the subproblem is integer, the optimal solution of the branch has been found. If a branch is fathomed because the value of the objective function of the optimal solution is worse than that of the incumbent solution, the branch does not contain the optimal solution of the original AWTINP. In any way, the optimal solution of the original AWTINP is not missed.

A BB tree is used to keep track of the branching variables and their bounds. One level is added to the BB tree each time the current subproblem is divided into two and one level is deleted from the BB tree each time when both branches have been fathomed. Two arrays $\mathbf{\Phi}$ and $\mathbf{C}$ are used to represent the BB tree. The branching variables are stored in $\mathbf{\Phi}$ and their bounds are stored in $\mathbf{C}$. An integer h is used to represent the level (or height or depth) of the BB tree and also to index the elements of $\mathbf{\Phi}$ and $\mathbf{C}$. For description purpose, each Φ_h is an ordered pair (p,q) or $(p,-q)$ indicating that the branching variable is x_{pq} at level h. The sign of q indicates which subproblem is currently examined. When the subproblem with $x_{pq} \leq \lfloor \tilde{x}_{pq} \rfloor$ is examined at level h, Φ_h stores $(p,-q)$ and C_h stores u_{pq}. When the BB tree is tracked back, $-q$ indicates that the branch with $x_{pq} \geq \lceil \tilde{x}_{pq} \rceil$ has not been examined. Because the upper bound on x_{pq} has been set to $\lfloor \tilde{x}_{pq} \rfloor$, u_{pq} is used to restore its original upper bound. When the subproblem with $x_{pq} \geq \lceil \tilde{x}_{pq} \rceil$ is examined, Φ_h stores (p,q) and C_h stores $\lceil \tilde{x}_{pq} \rceil$. When the BB tree is tracked back, q indicates that both branches on the branching variable x_{pq} have been fathomed. Because

the upper bound on x'_{pq} has been set to $u_{pq} - \lceil \tilde{x}_{pq} \rceil$ for the branch with $x_{pq} \geq \lceil \tilde{x}_{pq} \rceil$, $\lceil \tilde{x}_{pq} \rceil$ stored in C_h is used to restore the original upper bound on x_{pq}. In the actual implementation, the arcs are numbered from 1 to a and only the arc number or its negative, instead of the pair (p,q) or $(p,-q)$, is stored in Φ_h.

After an AWTNP is solved, the BB algorithm compares its optimal objective function value d with that of the incumbent solution d^{best}. If $d \geq d^{best}$, the current branch is fathomed. Otherwise, the algorithm searches for a non-integer non-key basic flow variable. Such a variable, if found, becomes the next branching variable and one level is added to the BB tree. If such a variable cannot be found, the current solution is an integer solution and the branch is fathomed. In this case, the current solution becomes the new incumbent solution.

If a branch is fathomed, the BB algorithm checks if both branches at the same level have been fathomed. If not, the subproblem with $x_{pq} \geq \lceil \tilde{x}_{pq} \rceil$ becomes the new subproblem to examine; otherwise, it tracks back one level and checks if both branches at this upper level have been examined. If not, the subproblem with $x_{pq} \geq \lceil \tilde{x}_{pq} \rceil$ at this upper level becomes the new subproblem to examine; otherwise, it tracks back one more level. After tracking back to the first level, the BB algorithm terminates. The following is a step by step description of the BB algorithm.

Step 1. Solve the initial AWTNP to obtain ($\tilde{\mathbf{x}}_\lambda$, $\tilde{\mathbf{z}}_\lambda$, $\tilde{\mathbf{s}}_\lambda$, $\tilde{\alpha}_\lambda$). Use the enumerative algorithm in Sun [2009] to find a supported integer efficient solution $\hat{\mathbf{x}}_\lambda$ with $\hat{\mathbf{z}}_\lambda = C\hat{\mathbf{x}}_\lambda$ and an objective value d. Let $\tilde{\mathbf{x}}_\lambda^{best} = \hat{\mathbf{x}}_\lambda$, $\tilde{\mathbf{z}}_\lambda^{best} = \hat{\mathbf{z}}_\lambda$, and $d^{best} = d$. Let $h = 0$. Let $k = 1$ and go to Step 2.

Step 2. If $k \leq K$, go to Step 3. If $k > K$, the current solution is an integer solution. Let $\tilde{\mathbf{x}}_\lambda^{best} = \tilde{\mathbf{x}}_\lambda$, $\tilde{\mathbf{z}}_\lambda^{best} = \tilde{\mathbf{z}}_\lambda$, and $d^{best} = d$. If $h = 0$, go to Step 9; otherwise, go to Step 6.

Step 3. If the k th non-key basic variable is a non-flow variable or is a flow variable but is integer, let $k = k+1$ and go to Step 2. Otherwise if the k th non-key basic variable is the flow variable x_{pq} with a fractional value $\tilde{x}_{pq}$, go to Step 4.

Step 4. Let $h = h+1$, $\Phi_h = (p,-q)$, $C_h = u_{pq}$ and $u_{pq} = \lfloor \tilde{x}_{pq} \rfloor$. Reduce x_{pq} to satisfy $x_{pq} \leq \lfloor \tilde{x}_{pq} \rfloor$ by making a basis exchange, and then go to Step 5.

Step 5. Solve the resulting AWTNP to obtain the optimal solution ($\tilde{\mathbf{x}}_\lambda$, $\tilde{\mathbf{z}}_\lambda$, $\tilde{\mathbf{s}}_\lambda$, $\tilde{\alpha}_\lambda$) with an objective value d. If $d > d^{best}$, go to Step 6; otherwise, let $k = 1$ and go to Step 2.

Step 6. If $\Phi_h = (p,-q)$, go to Step 7; otherwise, go to Step 8.

Step 7. Let $\Phi_h = (p,q)$, $t = C_h$, $C_h = u_{pq} + 1$, and $u_{pq} = t$. Increase x_{pq} by making a basis exchange. Let $u_{pq} = u_{pq} - C_h$, $b_p = b_p - C_h$, $b_q = b_q + C_h$, and $\mathbf{z}^{**} = \mathbf{z}^{**} - C_h * \mathbf{c}_{pq}$. If x_{pq} is basic, let $x_{pq} = x_{pq} - C_h$. If the resulting solution is infeasible, *i.e.*, if $x_{pq} < 0$, restore feasibility by making basis exchanges to let x_{pq} increase. Go to Step 5.

Step 8. Let $u_{pq} = u_{pq} + C_h$, $b_p = b_p + C_h$, $b_q = b_q - C_h$, and $\mathbf{z}^{**} = \mathbf{z}^{**} + C_h * \mathbf{c}_{pq}$. If x_{pq} is basic, let $x_{pq} = x_{pq} + C_h$. If x_{pq} is at its current lower bound, let $x_{pq} = C_h$ and make a basis exchange to make the resulting solution basic. Let $h = h - 1$. If $h = 0$, go to Step 9; otherwise, go to Step 6.

Step 9. Output $\tilde{\mathbf{x}}_\lambda^{best}$ and $\tilde{\mathbf{z}}_\lambda^{best}$, and Stop.

In Step 1, the initial AWTNP is solved and an initial incumbent solution is found. Step 2 checks if the current solution is an integer solution. If so, it is saved as the new incumbent solution. In Step 3, a non-integer non-key basic flow variable, if found, becomes the new branching variable. The subproblem with $x_{pq} \le \lfloor \tilde{x}_{pq} \rfloor$ is formed and one level is added to the BB tree in Step 4. When a basis exchange is made, (p,q) is the entering arc and x_{pq} is reduced. In Step 5, an AWTNP is solved and the branch is fathomed if $d > d^{best}$. Step 6 checks if both subproblems at the same level of the BB tree have been examined.

In Step 7, the AWTNP with $x_{pq} \ge \lceil \tilde{x}_{pq} \rceil$ in (12)-(20) is constructed after a basis exchange is made to increase x_{pq}. In this step, u_{pq}, b_p, b_q and $\mathbf{z}^{**}$ are reset. If (p,q) is currently basic, whether key or non-key, x_{pq} is reset by subtracting $C_h = \lceil \tilde{x}_{pq} \rceil$ from it. Otherwise, the change in x_{pq} is reflected in the changes of its bounds. More basis exchanges are made if needed to satisfy $x_{pq} \ge \lceil \tilde{x}_{pq} \rceil$.

The algorithm tracks one level back on the BB tree in Step 8. The originals of u_{pq}, b_p, b_q and $\mathbf{z}^{**}$ are restored. If (p,q) is currently basic, x_{pq} is reset by adding $C_h = \lceil \tilde{x}_{pq} \rceil$ to it. If x_{pq} is at its upper bound, this change is reflected in the change of its upper bound. If x_{pq} is at its lower bound, then $C_h = \lceil \tilde{x}_{pq} \rceil$ is added to it to reflect its actual value. However, this change makes the current solution non-basic. Hence, a basis exchange is made to find a basic solution. In this basis exchange, (p,q) is the entering arc and x_{pq} is reduced.

In an interactive procedure, many integer efficient solutions need to be sampled and, therefore, many AWTINPs with different $\boldsymbol{\lambda} \in \Lambda$ need to be formed and solved one after another. For each $\boldsymbol{\lambda} \in \Lambda$, the warm start routine [Sun, 2005a] is used to find an initial basic solution and the procedures in Sun [2003] are then used to find the best basic solution of the MONP problem. The special simplex method for network problems with side constraints is then used to find an optimal efficient solution of the initial AWTNP. The enumerative algorithm [Sun, 2009] is used to find a supported integer efficient solution as the initial incumbent solution. The BB algorithm is then used to find the optimal solution of the AWTINP. After the BB algorithm starts, all subsequent AWTNPs are solved with the special simplex method for network problem with side constraints.

5. An Example

A simple network with $n = 4$, $a = 5$ and $K = 2$ is shown in Figure 1. The b_j s are shown by the nodes and the c_{ij}^k for $k = 1$ and $k = 2$ are shown on the arcs. Although X and X^I cannot be shown graphically because of

their dimensionality, Z and Z^I are shown in Figure 2 where Z is shown as the shaded polygon and each $\mathbf{z} \in Z^I$ is shown as a dot.

In the MONP problem, solutions A, B, C and D are basic nondominated and the nondominated set Z is the union of the three line segments from solution A to solution B, from solution B to solution C and from solution C to solution D. All other feasible solutions are dominated by some others. In the MOINP problem, Z^I consists of the 11 feasible solutions as shown in Figure 2. All these feasible solutions are nondominated, *i.e.*, $\mathrm{Z}^I = Z^I$ and $\mathrm{X}^I = X^I$. All $\mathbf{x} \in X^I$ ($\mathbf{x} \in \mathrm{X}^I$) and $\mathbf{z} \in Z^I$ ($\mathbf{z} \in \mathrm{Z}^I$) are presented in Table 1.

The set $Z^{\geq}$ is shown in Figure 3. All $\mathbf{z} \in \mathrm{Z}$ are supported. Solutions A, B, C and D in Z^I are basic supported. Solutions H and K in Z^I are non-basic supported. Although solution E in Z^I is basic in the MONP problem and solutions E, G and J are supported in set theory, they are unsupported in the MOINP problem because they are not on the boundary of $Z^{\geq}$. These solutions cannot be reached by an AWTNP formed with the original MONP problem with any $\boldsymbol{\lambda} \in \Lambda$ because they are supported by a facet that is dominated in the MONP problem. Hence, based on the definition of supportedness [Steuer, 1986], solutions E, F, G, I, and J are unsupported in the MOINP problem.

Figures 1-3 and Table 1 approximately here

For this MONP problem, $\mathbf{z}^* = (0, -4)$. With $\mathbf{z}^{**} = (-1, -5)$ and $\rho = 0.001$, solution G is the integer optimal solution of the AWTINP for the weighting vector $\boldsymbol{\lambda} = (0.45, 0.55)$. Because solution G is unsupported, it cannot be reached with the AWTNP formed with $\boldsymbol{\lambda} = (0.45, 0.55)$. The optimal solution of the AWTNP is $\mathbf{x}_\lambda = (x_{12}, x_{13}, x_{14}, x_{23}, x_{24}) = (0.71028, 1.28972, 2.00000, 1.71028, 0.00000)$ with $\mathbf{z}_\lambda = (z_1, z_2) = (8.9720, 3.1589)$. In this solution, (1, 4) is an essential key basic arc, (1, 2) and (1, 3) are inessential key basic arcs, (2, 3) is a non-key basic arc and (2, 4) is a non-basic arc. This solution is shown in the key tree in Figure 4. In the figure, the flows are shown on the arcs. Adjusting the flows to integers, the enumerative algorithm of Sun [2009] found solution H, rather than solution G, as the best supported integer efficient solution.

Because $\mathbf{x}_\lambda$ is not an integer solution, the BB algorithm is used to find an optimal solution for the AWTINP. Solution H found with the enumerative algorithm of Sun [2009] is the initial incumbent solution with an objective function value $d = 5.419$.

After setting the new bound $x_{23} \leq 1$ in one subproblem, the reduced feasible region in criterion space is shown in Figure 5. Solution G becomes a nondominated basic solution of the new MONP problem and is the optimal solution of the new AWTNP. Because the optimal solution is integer, the branch with $x_{23} \leq 1$ is fathomed. Solution G becomes the new incumbent solution with an objective function value $d = 4.970$.

By setting $x_{23} \geq 2$ in the other subproblem, the reduced feasible region in criterion space along with $\mathbf{z}^{**}$ is shown in Figure 6. By shifting both Z and $\mathbf{z}^{**}$ to the left by $\mathbf{c}_{23}\lceil \tilde{x}_{23} \rceil = (4,0)2 = (8,0)$, the reduced feasible region along with $\mathbf{z}'^{**}$ of the actual subproblem solved is shown in Figure 7. Solution H becomes a nondominated basic solution of the new MONP problem and is the optimal solution of the new AWTNP with an objective value

$d = 5.419$. Hence, the branch with $x_{23} \geq 2$ is fathomed because its objective value is larger than that of the current incumbent solution. Even if its objective function value were not larger than that of the current incumbent solution, this branch could also be fathomed because it is an integer solution. Although solution H is an integer solution, it cannot become the new incumbent solution because it is not better than the current incumbent solution. Because both branches are fathomed, the problem is solved with the current incumbent solution G as the optimal solution.

Figures 4-7 approximately here

Larger MONP problems cannot be solved so easily. Usually each subproblem is further divided into two more subproblems. This division may go very deep before a branch is fathomed.

6. Computational Experience

The BB algorithm is coded in FORTRAN. The FORTRAN code is run on a Sun Enterprise 450 computer with two 400 MHz processors (only one is used) and 1.5 GB RAM. Other related algorithms used in the BB algorithm, such as the network simplex algorithm, the special simplex method for network problems with side constraints, and the warm start routines, are all coded in FORTRAN. All the computational results reported in this section represent the performance of the BB algorithm on this computer.

Test problems are measured by their sizes and the total supplies or demands. The size of each problem is measured by K and $n \times a$. Three sets of problems with $n \times a = 20 \times 100$, $n \times a = 30 \times 150$ and $n \times a = 40 \times 200$, respectively, are used. Among the n nodes in a test problem, $n/2$ are supply nodes and the other $n/2$ are demand nodes. The computational results of these three sets of problems are reported in Tables 2, 3 and 4, respectively. For each set of problems, $K = 3$, $K = 5$ and $K = 7$, respectively, are used. The total supply equals the total demand for each test problem. This total supply or total demand is denoted by B with $B = \sum_{\{i|b_i>0\}} b_i = -\sum_{\{i|b_i<0\}} b_i$. For each given K and given $n \times a$, three different values of B are used. Each combination of $n \times a$, K and B defines a problem type. Within each problem type, 10 test problems are used. These test problems are generated with the NP problem generator NETGEN [Klingman, Napier and Stutz, 1974]. NETGEN generates NP problems with a single objective function. Additional objective functions generated in the same fashion are added. For all these test problems, $1 \leq c_{ij}^k \leq 10$ for all $k = 1, \cdots, K$ and $(i, j) \in \mathrm{A}$ is used. For each test problem, 100 weighting vectors are used to form 100 AWTINPs to generate integer efficient solutions. The generation of 100 sample solutions may represent the computational effort needed to solve a typical MOP problem with an interactive procedure.

Because the BB algorithm is designed to find representative integer efficient solutions to support interactive procedures, it does not systematically enumerate the complete set of all integer efficient solutions for a MOINP problem. Criteria, such as the numbers of supported and unsupported integer efficient solutions in a MOINP problem used by Sedeño-Noda and González-Martín [2001] or the supported integer efficient solutions as a proportion of all integer efficient solutions used by Raith and Ehrgott [2009], can not be used in this study to measure the property of the test problems or the performance of the BB algorithm.

In Tables 2, 3 and 4, the total CPU time needed for all 100 AWTINPs, *i.e.*, for the generation of 100 integer efficient solutions, for each test problem is reported in column t. The level of the tallest BB tree among the 100

AWTINPs is reported in column $\hat{h}$. The number of subproblems solved for each AWTINP is recorded. The average and the maximum number of subproblems solved among the 100 AWTINPs for each test problem are reported in columns $\bar{s}$ and $\hat{s}$, respectively. The average of t and $\bar{s}$ of the 10 test problems in each problem type are also reported in these tables.

Table 2-4 approximately here

For any given $n \times a$ and given B, the test problems become more time consuming to solve as K increases. For the same network problem, the size of the nondominated set increases as K increases [Steuer, 1986; Sun, 2005b]. For the MOINP problem, the number of integer efficient solutions increases as K increases. Because there are more integer efficient solutions to check, the problem becomes more difficult to solve. When K increases, the number of side constraints (7) also increases. Therefore, the effort needed to solve each subproblem also increases. With one exception, the average of t all increased as K increased. For the $n \times a = 40 \times 200$ problems with $B = 800$, the average of t decreased from 63110 seconds to 28950 seconds when K increased from 5 to 7 because one problem with $K = 5$ took excessive amount of CPU time.

For any given $n \times a$ and given K, the problems become more difficult to solve as the values of B increase. When the values of B increase, the flows on the arcs also increase. As a result, there are more values to check for each integer variable and, therefore, more CPU time is needed. For all combinations of $n \times a$ and K, the average of t increased as B increased.

The computational effort is also affected by $n \times a$. When n increases, the size of T increases and, therefore, more time is needed to update T at each iteration of the network algorithm and of the special simplex method for network problems with side constraints. When a increases, the number of integer variables increases and more integer variables become candidates for the branching variables. Furthermore, the number of non-basic arcs increases and more computational effort is needed to find an optimal solution for each subproblem.

As with other integer programming problems, there is a big variation in the effort needed to solve the same type of MOINP problems. Within each problem type, there are big differences in both the CPU times taken and the numbers of subproblems solved. As expected, the CPU time taken is positively correlated to the number of subproblems solved. The level of the tallest BB tree does not vary much within each problem type.

As a well known fact, integer programming problems are very difficult to solve. Multiple objective integer programming problems are more difficult than single objective integer programming problems. Fortunately, MOINP problems are easier than other types of multiple objective integer programming problems because AWTNPs are easier to solve than many other types of mathematical programming problems. The BB algorithm is able to solve $n \times a = 40 \times 400$ problems within a reasonable amount of CPU time without using a powerful computer. Flow variables in MOINP problems are general integer variables, *i.e.*, they may take on any nonnegative integer values. Therefore, a also represents the number of general integer variables in a problem. Integer programming problems with general integer variables are more difficult than those with only binary variables because each general integer variable has more integer values to check. Therefore, this BB algorithm for the MOINP problems is considered to be very powerful.

7. Conclusions

A BB algorithm is developed, implemented and tested to generate representative integer efficient solutions for MOINP problems by solving AWTINPs. An AWTINP is solved by solving a sequence of AWTNPs in the BB algorithm. The BB algorithm is not expected to generate the whole set of integer efficient solutions but is to support interactive procedures by generating representative integer efficient solutions. Although computationally efficient network algorithms cannot be directly applied to the AWTNP, the BB algorithm takes advantage of the network structure as much as possible. The algorithm can generate basic and non-basic, as well as supported and unsupported, integer efficient solutions. Computational results show that the algorithm can solve pretty large problems within a reasonable amount of CPU time on a not very powerful computer. The number of arcs in a test problem represents the number of general integer variables. Hence, these test problems are considered pretty large in the integer programming context. Therefore, the BB algorithm is considered to be very powerful.

The supported integer efficient solutions found with the enumerative algorithm of Sun [2009] are sufficient to support interactive procedures for most applications. For some applications, especially when the flow variables have relatively small values, the BB algorithm is needed to generate both supported and unsupported integer efficient solutions.

Acknowledgement

This research was supported in part by a grant from the College of Business at University of Texas at San Antonio.

References

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

Bazaraa, M. S., J. J. Jarvis and H. D. Sherali, *Linear Programming and Network Flows*, Second Edition, Wiley, New York, New York, 1990.

Chen, S. and R. Saigal, "A Primal Algorithm for Solving a Capacitated Network Flow Problem with Additional Linear Constraints," *Networks*, Vol. 7, No. 1, pp. 59-79, 1977.

Current, J. and M. Marsh, "Multiobjective Transportation Network Design and Routing Problems: Taxonomy and Annotation," *European Journal of Operational Research*, Vol. 65, No. 1, pp. 4-19, 1993.

Current, J. and H. Min, "Multiobjective Design of Transportation Networks: Taxonomy and Annotation," *European Journal of Operational Research*, Vol. 26, No. 2, pp. 187-201, 1986.

Drinka, D., M. Sun and B. Murray, "A Multiple Objective Embedded Network Model for Human Resource Planning and an Implementation of the Tchebycheff Method," *Decision Sciences*, Vol. 27, No. 2, pp. 319-341, 1996.

Ehrgott, M. and X. Gandibleux, "A Survey and Annotated Bibliography of Multiobjective Combinatorial Optimization," *OR Spektrum*, Vol. 22, pp. 425–460, 2000.

Eusébio, A. and J. R. Figueira, "On the Computation of All Supported Efficient Solutions in Muti-Objective Integer Network Flow Problems," *European Journal of Operational Research*, Vol. 199, No. 1, pp. 68-76, 2009a.

Eusébio, A. and J. R. Figueira, "Finding Non-Dominated Solutions in Bi-Objective Integer Network Flow Problems," *Computers and Operations Research*, Vol. 36, No. 9, pp. 2554-2564, 2009b.

Glover, F., D. Klingman and N. V. Phillips, *Network Models in Optimization and Their Applications in Practice*, John Wiley and Sons, Inc., New York, 1992.

Hamacher, H. W., C. R. Pedersen and S. Ruzika, "Multiple Objective Minimum Cost Flow Problems: A Review," *European Journal of Operational Research*, Vol. 176, No. 3, pp. 1404-1422, 2007.

Isermann, H., "The Enumeration of the Set of All Efficient Solutions for a Linear Multiple Objective Program," *Operational Research Quarterly*, Vol. 28, No. 3, pp. 711-725, 1977.

Isermann, H., "Operating Manual for the EFFACET Multiple Objective Linear Programming Package," Fakultät für Wirtschaftswissenschaften, Universität Bielefeld, Germany, 1984.

Kennington, J. L. and R. V. Helgason, *Algorithms for Network Programming*, John Wiley and Sons, Inc., New York, New York, 1980.

Klingman, D., Napier, A., Stutz, J., "NETGEN - A Program for Generating Large Scale (Un)Capacitated Assignment, Transportation and Minimum Cost Network Problems, " *Management Science*, Vol. 20, No. 5, pp. 814-822, 1974.

Lee, H. and P. S. Pulat, "Bicriteria Network Flow Problems: Integer Case," *European Journal of Operational Research*, Vol. 66, No.1, pp. 148-157, 1993.

Murty, K. G., *Network Programming*, Prentice Hall, Englewood Cliffs, New Jersey, 1992.

Mustafa, A. and M. Goh, "Finding Integer Efficient Solutions for Bicriteria and Tricriteria Network Flow Problems Using DINAS," *Computers & Operations Research*, Vol. 25, No. 2, pp. 139-157, 1998.

Nemhauser, G. L. and L. A. Wolsey, *Integer and Combinatorial Optimization*, Wiley, New York, New York, 1988.

Ogryczak, W., K. Studzi·ski and K. Zorychta, "A Solver for the Multi-objective Transshipment Problem with Facility Location," *European Journal of Operational Research*, Vol. 43, No. 1, pp. 53-64, 1989.

Ogryczak, W., K. Studzi·ski and K. Zorychta, "DINAS: A Computer-Assisted Analysis System for Multiobjective Transshipment Problems with Facility Location," *Computers & Operations Research*, Vol. 19, No. 7, pp. 637-647, 1992.

Özlen, M. and M. Azizoğlu, "Multi-objective Integer Programming: A General Approach for Generating All Non-dominated Solutions," *European Journal of Operational Research*, Vol. 199, No. 1, pp. 25-35, 2009.

Przybylski, A., X. Gandibleux and M. Ehrgott, "The Biobjective Integer Minimum Cost Flow Problem– Incorrectness of Sedeño-Noda and González-Martín's Algorithm," *Computers & Operations Research*, Vol. 33, No. 5, pp. 1459–63, 2006.

Przybylski, A., X. Gandibleux and M. Ehrgott, "A Recursive Algorithm for Finding All Nondominated Extreme Points in the Outcome Set of a Multiobjective Integer Programme," *INFORMS Journal on Computing*, forthcoming, 2009.

Pulat P.S., F. Huarng, and H. Lee, "Efficient Solutions for the Bicriteria Network Flow Problem," *European Journal of Operational Research*, Vol. 19, No. 7, pp. 649–55, 1992.

Raith, A. and M. Ehrgott, "A Two-Phase Algorithm for the Biobjective Integer Minimum Cost Flow Problem," *Computers and Operations Research*, Vol 36, No. 6, pp. 1945-1954, 2009.

Ruhe, G., "Complexity Results for Multicriterial and Parametric Network Flows Using a Pathological Graph of Zadeh," *Zeitschrift für Operations Research*, Vol. 32, No. 1, pp. 9-27, 1988.

Sedeño-Noda, A. and C. González-Martín, "The Biobjective Minimum Cost Flow Problem," *European Journal of Operational Research*, Vol. 124, No. 3, pp.591–600, 2000.

Sedeño-Noda, A. and C. González-Martín, "An Algorithm for the Biobjective Integer Minimum Cost Flow Problem," *Computers & Operations Research*, Vol. 28, No. 2, pp. 139-156, 2001.

Sedeño-Noda, A. and C. González-Martín, "An Alternative Method to Solve the Biobjective Minimum Cost Flow Problem," *Asia-Pacific Journal of Operational Research*, Vol. 20, pp. 241–60, 2003.

Steuer, R. E., ADBASE: A Multiple Objective Linear Programming Solver for All Efficient Extreme Points and All Unbounded Efficient Edges, Terry College of Business, University of Georgia, Athens, GA, 2003.

Steuer, R. E., *Multiple Criteria Optimization: Theory, Computation, and Application*, John Wiley and Sons, Inc., New York, New York, 1986.

Steuer, R. E. and E.-U. Choo, "An Interactive Weighted Tchebycheff Procedure for Multiple Objective Programming," *Mathematical Programming*, Vol. 26, No. 1, pp. 326-344, 1983.

Steuer, R. E. and C. Piercy, "Difficulties in Solving Network Multiple Objective Linear Programming Problems," in Lawrence, K. D. (ed.), *Multi-Criteria Applications, Applications of Management Science*, Vol. 10, Elsevier Science, pp. 217-231, 2000.

Steuer, R. E., J. Silverman and A. W. Whisman, "A Combined Tchebycheff /Aspiration Criterion Vector Interactive Multiobjective Programming Procedure," *Management Science*, Vol. 39, No. 10, pp. 1255-1260, 1993.

Sun, M., "Procedures for Finding Nondominated Solutions for Multiple Objective Network Programming Problems," *Transportation Science*, Vol. 37, No. 2, pp. 139-152, 2003.

Sun, M., "Warm-Start Routines for Solving Augmented Weighted Tchebycheff Programs in Multiple Objective Network Programming," *INFORMS Journal on Computing*, Vol. 17, No. 4, pp. 422-437, 2005a.

Sun, M., "Some Issues in Measuring and Reporting Solution Quality of Interactive Multiple Objective Programming Procedures," *European Journal of Operational Research*, Vol. 162, No. 2, pp. 468-483, 2005b.

Sun, M., "Finding Integer Efficient Solutions for Multiple Objective Network Programming Problems," Department of Management Science and Statistics, The University of Texas at San Antonio, San Antonio, TX 78249, 2009.

Sun, M., A. Stam, and R. E. Steuer, "Solving Multiple Objective Programming Problems Using Feed-forward Artificial Neural Networks: The Interactive FFANN Procedure," *Management Science*, Vol. 42, No. 6, pp. 835-849, 1996.

Sun, M., A. Stam, and R. E. Steuer, "Interactive Multiple Objective Programming Using Tchebycheff Programs and Artificial Neural Networks," *Computers & Operations Research*, Vol. 27, No. 7-8, 601-620, 2000.

Table 1. Integer Efficient Solutions of the Example Network Problem

Solution	x_{12}	x_{13}	x_{14}	x_{23}	x_{24}	z_1	z_2
A	0	3	1	0	1	0	9
B	0	2	2	1	0	4	6
C	2	0	2	3	0	18	–2
D	4	0	0	3	2	24	–4
E	1	3	0	0	2	3	8
F	1	2	1	1	1	7	5
G	2	2	0	1	2	10	4
H	1	1	2	2	0	11	2
I	2	1	1	2	1	14	1
J	3	1	0	2	3	17	0
K	3	0	1	3	1	21	–3

Table 2. Computational Results of $n \times a = 20 \times 100$ problems

K	prob	$B = 50$				$B = 100$				$B = 200$			
		t	$\hat{h}$	$\bar{s}$	$\hat{s}$	t	$\hat{h}$	$\bar{s}$	$\hat{s}$	t	$\hat{h}$	$\bar{s}$	$\hat{s}$
3	1	2.37	52	163	1162	4.20	51	276	1810	6.29	50	405	3360
	2	2.76	54	191	1312	6.49	49	471	7810	10.23	49	729	9584
	3	2.80	46	195	964	4.07	55	284	1618	5.00	52	333	2024
	4	4.51	48	293	2554	8.68	48	604	5454	11.53	52	777	9820
	5	1.35	40	101	462	3.17	52	231	1488	7.62	48	545	8428
	6	3.38	50	213	1544	4.86	59	329	1956	11.41	65	733	5378
	7	3.88	53	242	1530	8.17	55	481	4854	12.31	60	702	3740
	8	1.71	41	130	922	3.48	50	255	2226	7.81	46	543	4400
	9	0.45	24	36	210	0.69	38	57	362	1.02	37	88	642
	10	3.68	50	234	2502	4.79	56	310	1554	10.89	58	651	9492
		2.69		180		4.86		330		8.41		550	
5	1	5.26	50	244	1688	8.40	55	397	3324	11.29	56	488	9756
	2	6.70	55	287	1718	13.63	59	559	3714	17.39	64	741	3704
	3	9.33	58	366	3762	13.93	60	555	5954	25.28	62	996	11246
	4	2.18	50	105	1170	3.73	60	186	1768	5.61	66	267	2204
	5	6.69	50	279	3160	16.07	55	658	7640	27.82	62	1105	21728
	6	11.65	56	421	12980	15.19	55	593	4478	25.21	58	1012	13816
	7	3.19	57	141	1372	8.78	68	363	5190	16.99	67	661	8066
	8	3.08	40	146	772	5.14	52	233	3268	8.76	57	404	2344
	9	10.89	67	453	5668	13.18	57	553	3772	25.55	60	1051	12280
	10	6.28	52	280	1428	14.57	60	611	6446	22.83	58	894	11368
		6.53		272		11.26		471		18.67		762	
7	1	9.77	57	295	4372	20.94	60	623	5446	31.70	61	954	7986
	2	6.80	51	234	1322	11.40	55	398	3832	20.28	49	636	4004
	3	7.01	53	236	1920	11.15	49	359	5546	30.76	57	900	17596
	4	8.50	50	256	3268	17.12	55	516	12258	26.34	55	813	13104
	5	4.83	45	164	1480	7.84	45	266	1754	12.28	49	428	3952
	6	7.27	52	252	1956	13.40	55	438	6348	20.21	52	708	6300
	7	9.08	51	288	2570	12.12	51	405	3392	24.33	63	765	10006
	8	3.36	48	125	1180	5.89	49	217	3374	6.56	43	256	2098
	9	9.78	51	302	2350	17.98	49	538	4954	27.38	51	834	14902
	10	10.55	50	331	7568	42.15	53	1176	69094	48.88	66	1412	64182
		7.69		248		16.00		494		24.87		771	

Table 3. Computational Results of $n \times a = 30 \times 150$ Problems

K	prob	$B = 100$				$B = 200$				$B = 400$			
		t	$\hat{h}$	$\bar{s}$	$\hat{s}$	t	$\hat{h}$	$\bar{s}$	$\hat{s}$	t	$\hat{h}$	$\bar{s}$	$\hat{s}$
3	1	15.40	74	560	8084	17.34	82	692	4302	48.81	79	2014	40270
	2	29.59	70	938	25930	47.82	77	1640	27940	93.93	79	3090	100926
	3	5.03	71	211	1592	10.13	68	430	3946	26.31	86	1016	11088
	4	16.79	80	572	4492	25.74	79	903	6808	43.58	75	1565	13420
	5	13.62	68	527	2970	27.46	71	1161	22610	29.49	77	1166	10654
	6	24.06	80	864	8046	40.66	71	1491	10076	169.66	94	7414	101624
	7	6.05	69	278	1468	9.31	64	474	3272	12.46	64	627	11794
	8	13.61	73	525	4110	19.24	85	765	6350	35.79	89	1484	20592
	9	12.53	74	499	2672	24.01	78	992	6920	45.92	81	1867	11590
	10	7.09	69	281	1780	16.20	75	692	3982	27.95	70	1324	10880
		14.38		525		23.79		924		53.39		2157	
5	1	77.26	86	1718	27220	125.13	83	2921	39764	329.96	94	6626	106468
	2	30.56	75	738	9642	48.46	80	1247	12674	75.05	80	1802	16470
	3	21.90	69	578	4628	49.06	72	1313	11684	78.96	89	2018	24992
	4	12.71	60	346	2296	20.07	67	597	3112	29.56	66	820	5174
	5	19.76	69	511	3616	34.27	74	872	8724	55.52	99	1385	13562
	6	38.60	77	964	8272	123.18	81	2754	30284	248.57	86	4667	101116
	7	32.51	93	788	6086	89.16	90	2119	17316	198.12	102	4378	53048
	8	38.20	75	920	10392	72.22	82	1691	18364	141.09	86	3421	32706
	9	40.02	88	980	7162	52.97	80	1412	11828	163.16	95	3987	29204
	10	56.53	86	1271	11694	87.96	87	2170	16638	164.49	91	3873	40174
		36.80		881		70.25		1710		148.45		3298	
7	1	66.09	73	1189	18154	128.54	83	2164	50728	219.22	98	3837	41768
	2	60.68	89	1159	7690	122.11	90	2197	19452	198.87	88	3585	35394
	3	42.77	82	779	15202	94.02	87	1606	23740	149.82	85	2560	29698
	4	73.43	80	1282	25420	126.32	90	2548	15662	937.04	90	23871	1358170
	5	51.63	69	1013	9572	181.26	79	2671	109068	253.18	88	4197	95168
	6	25.98	73	528	5622	71.66	75	1402	17572	133.76	83	2538	31114
	7	37.02	71	727	6342	76.58	76	1559	15412	147.52	87	2645	37358
	8	27.99	70	567	5432	67.06	85	1379	16088	94.62	80	1915	13650
	9	32.47	80	646	7806	94.85	91	1734	24482	227.17	91	3827	38510
	10	31.42	72	649	6632	57.80	72	1049	21086	106.04	91	1981	31348
		44.95		854		102.02		1831		246.73		5096	

Table 4. Computational Results of $n \times a = 40 \times 400$ Problems

K	prob	$B = 200$				$B = 400$				$B = 800$			
		t	$\hat{h}$	$\bar{s}$	$\hat{s}$	t	$\hat{h}$	$\bar{s}$	$\hat{s}$	t	$\hat{h}$	$\bar{s}$	$\hat{s}$
3	1	394.77	185	6959	190942	614.95	205	11839	120032	1363.94	201	26490	484886
	2	1094.16	240	15714	266798	3756.15	243	54975	851372	8796.35	246	140209	2623836
	3	1128.71	214	18838	240480	1686.14	218	29088	316280	42354.23	220	915990	47226968
	4	865.46	227	14306	208112	14156.58	224	280929	9723024	8847.93	250	128086	7494238
	5	274.66	209	4320	40430	898.13	214	15941	312884	2811.18	226	49452	1004072
	6	490.36	230	7622	71170	1662.50	263	29760	1428798	3444.54	250	57409	1615852
	7	665.55	221	11850	88906	1067.12	234	20315	341074	3799.15	227	70550	2039880
	8	336.27	199	5730	36242	834.22	232	13222	110532	2894.54	226	47414	875562
	9	422.19	233	6875	123802	1239.51	235	20525	257462	13387.11	253	199218	17493962
	10	466.14	231	6809	48870	1548.30	249	21416	388524	10541.35	255	170657	2899046
		613.83		9902		2746.36		49801		9824.03		180547	
5	1	1335.98	225	13902	136202	7166.15	243	97396	6362276	31231.40	233	470470	29378042
	2	800.89	235	8235	174388	2683.52	244	28058	368430	7279.40	237	64641	2565176
	3	1571.40	262	15770	157048	2500.44	295	24697	220110	8390.20	255	74422	2025720
	4	2517.64	270	25346	327288	3120.17	233	29980	425142	98885.68	253	1618440	154657844
	5	3908.76	263	36189	799442	6161.57	235	66336	1104848	432597.63	268	6465340	623178056
	6	867.52	240	9155	105720	2357.85	222	29766	760208	13584.18	236	197323	14826070
	7	2028.86	241	18430	403678	3183.82	261	30122	887408	4757.11	254	51411	564982
	8	2447.97	245	22346	589074	4748.46	224	42544	462294	6901.99	253	72337	810746
	9	2894.67	226	25149	1050972	5449.91	231	45958	2064114	12418.61	240	116760	3337962
	10	950.83	242	10049	90956	2539.20	241	26568	245746	15054.36	243	220576	13597054
		1932.45		18457		3991.11		42142		63110.06		877385	
7	1	2024.55	244	15492	205530	4299.90	287	31155	741846	10711.10	234	76658	1921120
	2	3161.18	240	23861	362288	47055.44	237	573433	46163362	12919.55	243	110302	2612912
	3	748.80	219	6459	63110	2897.35	242	19937	472780	5442.28	250	40270	634354
	4	2509.27	233	18885	243830	10159.68	235	80049	1960620	11087.77	266	80660	1495602
	5	1828.61	248	13558	191778	4690.36	237	42806	1001890	22649.81	252	241745	14525142
	6	2901.93	246	22289	350284	8148.64	256	57332	736318	14080.63	260	104524	980652
	7	2700.52	231	20418	356464	6387.17	252	47401	1641964	186109.48	267	2661868	259072094
	8	3292.61	232	23724	472814	10399.10	239	70995	2177720	15817.57	242	110406	1329994
	9	1115.98	200	9178	228528	1540.09	215	14447	130998	4225.35	220	35676	351958
	10	1054.65	207	8514	95308	2638.73	239	21158	203934	6459.98	274	61288	2333234
		2133.81		16238		9821.64		95871		28950.35		352340	

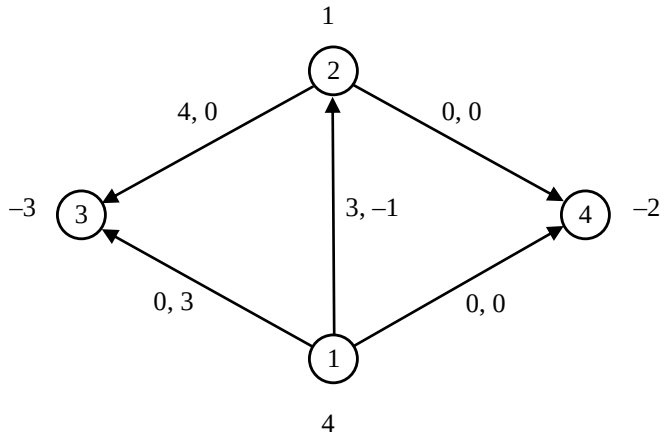

Figure 1. A Simple Network Problem

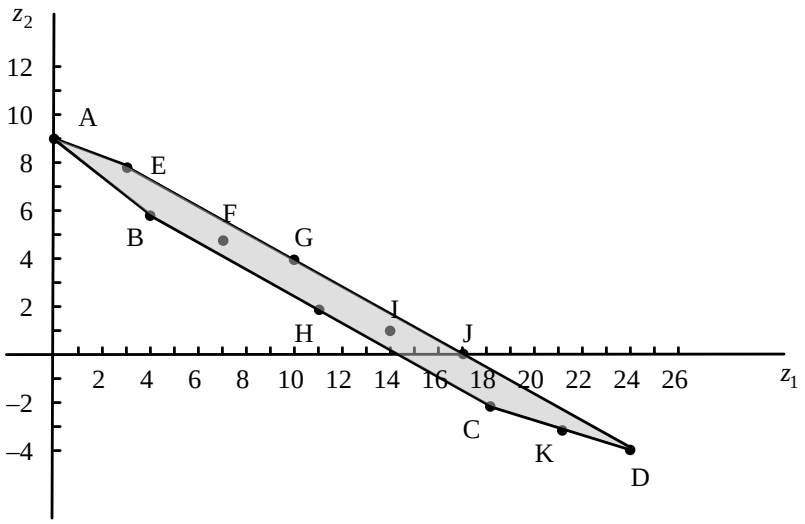

Figure 2. The Feasible Region in Criterion Space of the Simple Network Problem

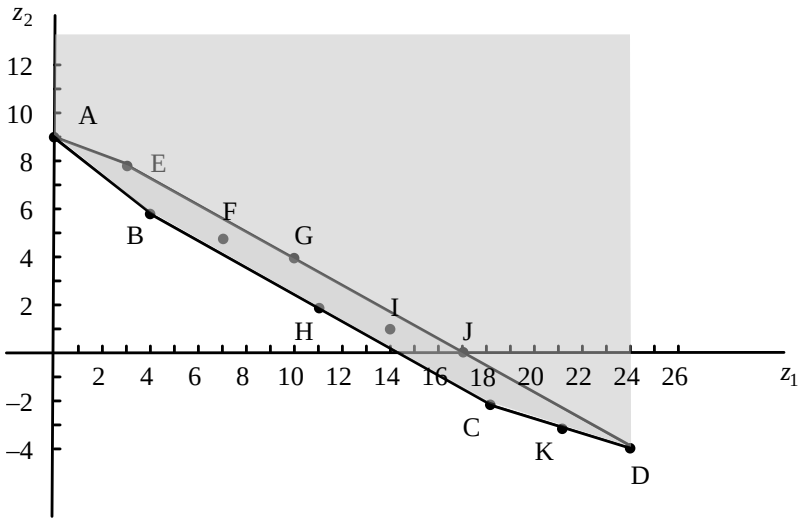

Figure 3. The Set $Z^{\geq}$ of the Simple MOINP Problem

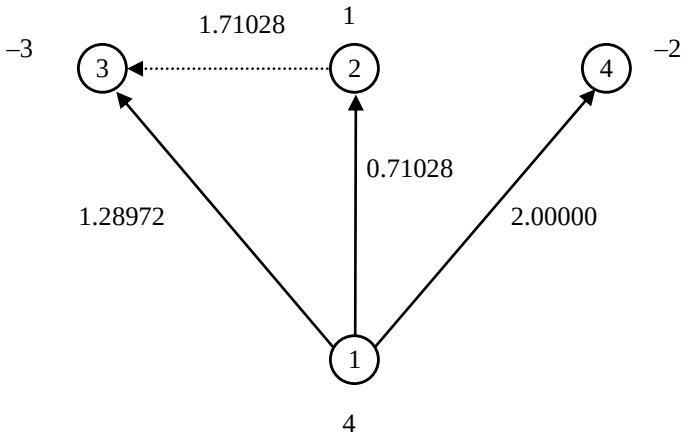

Figure 4. The Optimal Solution of the AWTNP

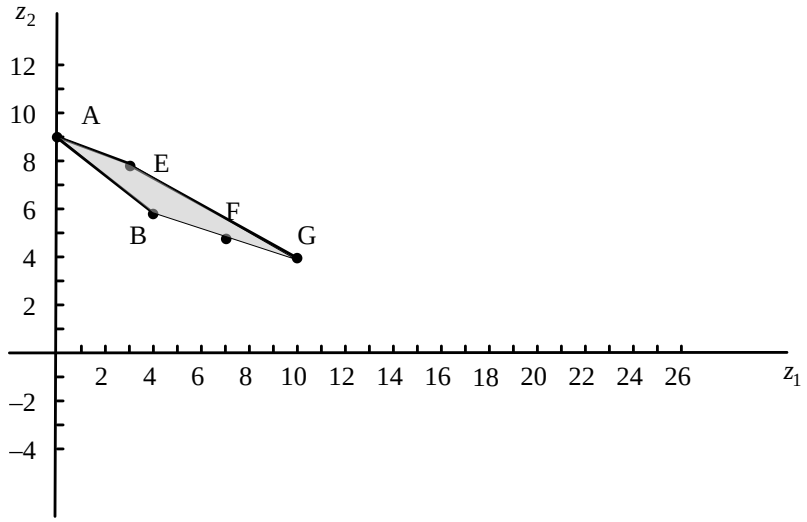

Figure 5. The Reduced Feasible Region in Criterion Space after Adding $x_{23} \leq 1$

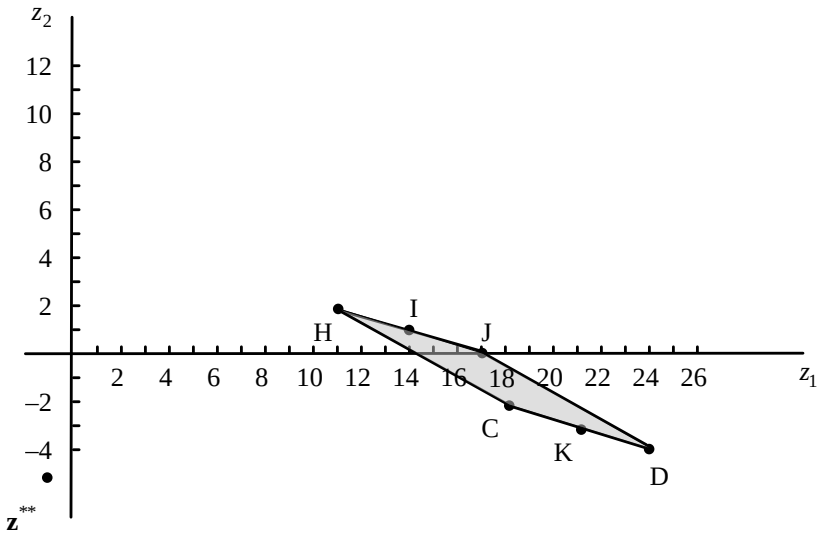

Figure 6. The Reduced Feasible Region in Criterion Space after Adding $x_{23} \geq 2$

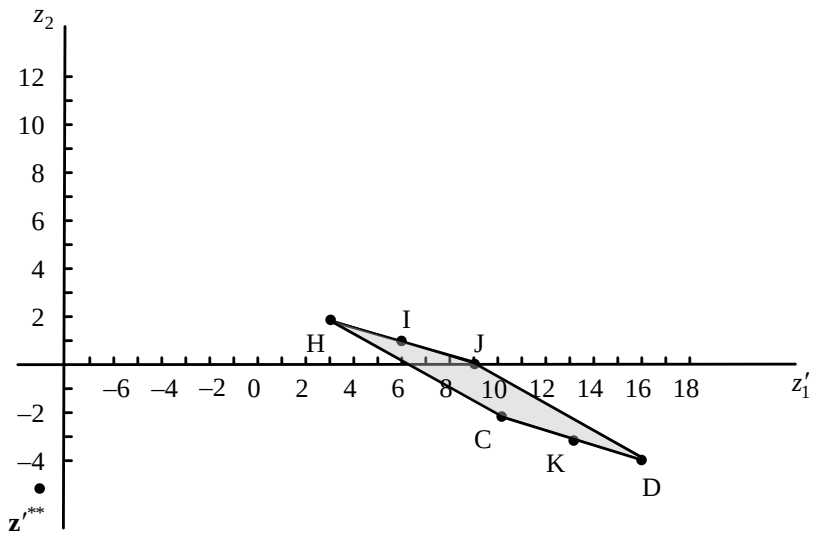

Figure 7. The Reduced Feasible Region in Criterion Space after Adding $x_{23} \geq 2$