TECHNICAL REPORT 04-EMIS-01

An Interval-Pivoting Algorithm for the Uniform-Bound Interval-Flow Transportation Problem

Aruna Apte* Richard S. Barr†

May 19, 2004

Abstract

Presented herein is a new class of network flow models, *interval-flow networks*, in which the flow on an arc may be required to be either zero or within a specified range. The addition of such conditional lower bounds creates a mixed-integer program that captures such well-known restrictions as minimum load sizes, minimum class enrollments, and minimum capacity utilization in telecommunications network spans.

This paper describes the mathematical properties of interval-flow networks as the basis for an efficient new heuristic approach that incorporates the conditional bounds into the simplex pivoting process and exploits the efficient, specialized pure-network simplex technologies. The algorithm is applied to interval-flow transportation problems with a uniform conditional lower bound and tested on problems with up to 5000 nodes and 10,000 arcs. Empirical comparisons with CPLEX demonstrate the effectiveness of this methodology, both in terms of solution quality and processing time.

*Cox School of Business, Southern Methodist University, Dallas, TX 75275. Email: aapte@cox.smu.edu.

†Department of Engineering Management, Information, and Systems, School of Engineering, Southern Methodist University, Dallas, TX 75275. Email: barr@engr.smu.edu. Sponsored in part by the THECB Advanced Technology Program grant 003613-023.

From its origins as one of the first applications of linear programming [10, 9], the transportation problem and its brethren network models—assignment, transshipment, generalized, and multicommodity—have found a broad range of applications. Similarly, research into specialized network optimization techniques have produced highly efficent algorithms and software that can effectively address practical, ultra-large-scale problems (e.g., [1]). Despite their popularity and widespread use, traditional models cannot address some situations with a dominant network structure because of the requirement of non-network constraints.

This work introduces an important new class of such problems, motivated by specific applications, that we call *interval-flow transportation networks*. In an interval-flow network, each arc may have a *conditional lower bound*, which restricts flow to be either zero or within a specified range. Depending on the application, the minimum (conditional) flow level may vary or be uniform across the designated arcs and typically reflects minimum utilization levels, such as load sizes, class enrollments, or traffic capacity. While this enhancement creates mixed-integer problems from the classical linear models, it also broadens their range of applications and adds greater realism.

For these computationally challenging problems, this paper presents an effective new type of heuristic that incorporates the interval-flow constraints into the simplex pivoting process and exploits the efficient, specialized simplex technologies that have been developed for pure networks. When embedded into a tabu search framework, the algorithm quickly provides high-quality solutions for interval-flow transportation networks.

The sections below formulate the uniform-bound interval-flow transportation problem, describe its mathematical properties, and develop an interval-pivoting algorithm for its solution. An implementation of this heuristic is detailed, and statistically designed series of computational experiments demonstrate the efficiency and quality of this approach.

1 Formulation and Properties

An interval-flow transportation problem with uniform lower bound, (UIFTP), can be formulated as the following mixed-integer model.

(P):

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} = z \tag{1}$$

$$s.t. \quad \sum_{(i,j)\in A} x_{ij} - \sum_{(k,i)\in A} x_{ki} = b_i, \forall i \in N \tag{2}$$

$$\ell y_{ij} \le x_{ij} \le \xi y_{ij},\ \forall (i,j) \in A \tag{3}$$

$$0 \le y_{ij} \le 1,\ \forall (i,j) \in A \tag{4}$$

$$y_{ij} \qquad \text{integer, } \forall (i,j) \in A \tag{5}$$

Each arc $(i,j) \in A$ is defined such that $i \in N_o$ and $j \in N_d$, where $N_o \cap N_d = \emptyset$, $N_o \cup N_d = N$, $b_i > 0 \ \forall i \in N_o$, and $b_j < 0 \ \forall j \in N_d$. The conditional lower bound is ℓ and $\xi = \sum_{i \in N_o} b_i$ is the total supply. In this model, ℓ is applied to all members of A and the relaxation of (5) results in $(\overline{P})$, an uncapacitated transportation problem.

The mixed-integer programming (MIP) nature of (P) puts UIFTP in class $\mathcal{NP}$-hard. Constraints (1) and (2) form the embedded network model. Interval flow transportation problem with uniform lower bound, as formulated in (P), have the following properties.

Property 1 *For a given arc (i,j) in (P), $y_{ij} = 0 \Longrightarrow x_{ij} = 0$ and $y_{ij} = 1 \Longrightarrow \ell \leq x_{ij} \leq \xi$.*

Proof: This follows from (3). ∎

Property 2 *If (P) is feasible, there exists an optimal solution that is a spanning tree for N.*

Proof: From (4) and (5), the optimal solution to a feasible (P) will contain $y_{ij} \in \{\ 0, 1\}$, $\forall (i,j) \in A$. Setting each y_{ij} to 0 or 1 reduces (P) to a pure network problem. We know that a pure network has a rooted spanning tree for its basis and that the set of optimal solutions for a feasible linear program will include a basic feasible solution [5]. Therefore, if an optimal solution for (P) exists, there is an optimal spanning tree solution. ∎

The feasible region of (P) is given by (2) – (5). Because of (3), (4), and (5), the disjunctive interval-flow constraints, the feasible region is a disjunction or union of convex feasible subregions, as illustrated in Figure 1. The following property is evident.

Property 3 *The feasible region of (P) is a disjunction of convex regions.*

Property 4 *The objective function for (P) is discontinuous.*

Proof: As a consequence of Property 3, the cost associated with arc (i,j) is 0 if $x_{ij} = 0$, is $c_{ij}x_{ij}$ for $x_{ij} \geq \ell$, and is undefined for $0 < x_{ij} < \ell$. ∎

These features of (P) are derived from the special structure of pure interval-flow networks and can be exploited for computational advantage. The algorithms described in the upcoming sections are built around these mathematical characteristics.

Property 5 *(P) is infeasible if there exists $0 <\mid b_i \mid < \ell$ for any $i \in N$.*

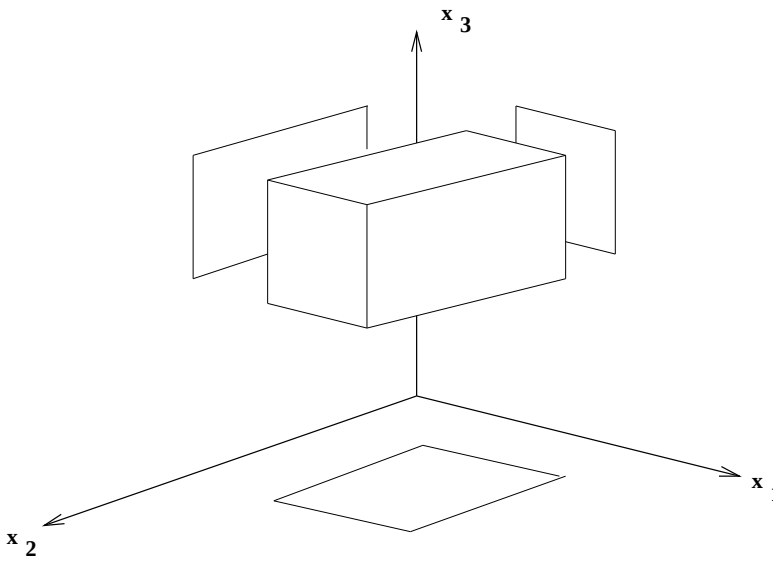

Figure 1: Nonconvex feasible region of UIFTP

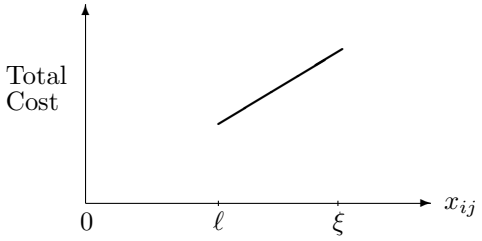

Figure 2: Nonconvex objective function for an UIFTP arc (i, j)

Proof: Let i be a node $\in N_o$ such that $0 < b_i < \ell$. For feasibility, x_{ij}, must satisfy constraints (2) and (3). But, since $\sum_{j \in N,\ (i,j) \in A} x_{ij} = b_i,\ \forall i \in N_o$, there must be some $(i,j) \in A$ such that $0 < x_{ij} < \ell$. This will violate (3). An analogous situation exists for any node $i \in N_d$ with $-\ell < b_i < 0$. Therefore (P) is infeasible if there exists $0 <\mid b_i \mid < \ell$. ∎

The following sections describe a new heuristic algorithm for (P) that exploit these properties, and compare an implementation of it with commercial-grade optimization software. These computational experiments demonstrate the strengths of the new approach on a broad-ranging suite of test problems.

2 Interval-Pivoting Algorithms

Presented below are a series of *interval-pivoting algorithms* (IPAs) for the UIFTP that capitalize on the properties 1–5. They are similar in philosophy to the adjacent-extreme-point heuristics proposed for fixed-charge [3, 4, 16, 17] and more general problems [4, 13], but use a unique representation for solutions of (P) that can be readily manipulated by modified network simplex procedures.

Any feasible solution of (P) will contain $y_{ij} \in \{0,1\},\ \forall (i,j) \in A$. Fixing each y_{ij} to 0 or 1 reduces (P) to a pure network problem. Constraint 4 ensures that a feasible solution to this fixed network will have $x_{ij} = 0$ or $\ell \leq x_{ij} \leq \xi, \forall (i,j) \in A$. A basic feasible solution to this fixed network will have $m-1$ basic variables that are linearly independent. The remaining $n+m-1$ variables are nonbasic. Such a basic feasible solution can be represented by the set $\{B, L, \Lambda\}$, where B is the set of x_{ij}s that are basic, L is the set of x_{ij}s that are nonbasic at 0, and Λ is the set of x_{ij}s that are nonbasic at ℓ. On the other hand, it is clear that, given such a $\{B, L, \Lambda\}$, we can calculate $x_{ij}, \forall (i,j) \in A$. Similarly, we can determine $y_{ij}, \forall (i,j) \in A$, based upon the x_{ij} values (e.g., if $x_{ij} = 0$, then the corresponding $y_{ij} = 0$, and if $x_{ij} \geq \ell$, then the corresponding $y_{ij} = 1$). Thus an interval-feasible basic solution to (P) can be represented by $\{B, L, \Lambda\}$, satisfies the network constraints (2)–(4), and implicitly accounts for the feasibility of y_{ij}s. We now offer formal definitions that form the foundation for the algorithms.

Definition 1 *An* interval-basic solution *(IBS) for (P) is a solution for which $n - m + 1$ (nonbasic) variables x_{ij} are equal to either their lower bound (0) or conditional lower bound (ℓ), and the remaining $m - 1$ (basic) variables are linearly independent. An IBS can be represented by the set $\{B, L, \Lambda\}$, where B, L, Λ are the sets of variables that are interval-basic (or IF-basic), IF-nonbasic at 0, and IF-nonbasic at ℓ, respectively. (Note that a basic solution is an IBS with $\Lambda = \emptyset$.)*

Definition 2 *A* network-feasible interval-basic solution *(NFIBS) is an IBS that satisfies constraints (2)–(4).*

Definition 3 *An* interval-feasible interval-basic solution *(IFIBS) is an IBS that satisfies all constraints of (P). Such a solution is said to be* IF-feasible*, as is any arc $(i, j) \in A$ satisfying constraints (3)–(5). All other IBSs and arcs are termed* IF-infeasible*.*

Definition 4 *Two IBS's, $\{B_1, L_1, \Lambda_1\}$ and $\{B_2, L_2, \Lambda_2\}$, are said to be* adjacent *if $|B_1 \cap (L_2 \cup \Lambda_2)| + |(L_1 \cap \Lambda_2) \cup (L_2 \cap \Lambda_1)| = 1$.*

Based on these definitions, we state another property of (P) which is extremely important in the construction of IPAs.

Property 6 *Depending on the status of y_{ij}, the corresponding x_{ij} variable can be nonbasic at 0, or ℓ.*

Proof: This follows from Property 1 and the above definitions. ∎

Just as the simplex method examines a series of adjacent basic feasible solutions to identify an optimal one, the interval-pivoting algorithms examine a succession of adjacent NFIBSs in their search for high-quality IF-feasible solutions. Since an interval-basic solution can be represented as a spanning tree, the algorithms can employ the data structures developed for network-simplex methods. Modified network-simplex pivots are used to move between adjacent NFIBSs, each time bringing a nonbasic arc into the solution and removing an interval-basic.

For these algorithms, the sections below provide detailed descriptions based on the following terms and the nomenclature of Ford and Fulkerson [6].

B	:	set of IF-basic arcs
L	:	set of IF-nonbasic arcs at flow of 0
Λ	:	set of IF-nonbasic arcs at flow of ℓ
B^c	:	set of IF-nonbasic arcs, $(L \cup \Lambda)$
$P(i, j)$	:	path from node i to node j in basis tree B [6]
$B_{bep}(i, j)$	:	set of IF-basic arcs in $P(j, i)$
$W(P(i, j))$	:	set of forward arcs in $P(i, j)$
$V(P(i, j))$	:	set of reverse arcs in $P(i, j)$
$A^+(i, j)$	:	$W(P(i, j)) \cup (i, j)$
$A^-(i, j)$	:	$V(P(i, j))$
F	:	set of IF-feasible arcs
F^c	:	set of IF-infeasible arcs
$d(i, j)$	:	$\max\{0, \min\{x_{ij}, \ell - x_{ij}\}\}$, for $(i, j) \in A$
$d(S)$	:	$\sum_{(i,j)\in S} d(i, j)$, for any nonempty set $S \subseteq A$

$$
\begin{aligned}
f(S) &: \mid S \cap F^c \mid \\
b_{min} &: \min_{i \in N} \{\mid b_i \mid\} \\
\delta &: \text{possible flow change}
\end{aligned}
$$

Guiding the IPA search are three metrics: $\overline{c}_{ij}$, the reduced cost of an arc (i, j), computed using the dual variables of the current NFIBS; $f(B)$, the number of IF-infeasible arcs in a given NFIBS, B; and $d(B)$, the *integer feasibility* of B [15]. The $d(B)$ values are determined from $d(i, j)$, the *distance from feasibility* for an IF-infeasible arc (i, j), defined as the smallest amount of flow change (increase or decrease) required to reach an IF-feasible state. These metrics assess improvement when evaluating the effects of a given change of IBS.

Because of discontinuities in the objective function and non-convexity of the feasible region of (P), evaluation of an IF-nonbasic arc involves the inspection of its basis-equivalent path. Traversal of BEP is required (sometimes repeatedly) to determine: the arc's impact on $d(B)$ and $f(B)$, the flow-change possibilities, and the corresponding arcs that would leave B. Evaluation complexity arises from the fact that, for a given NFIBS and incoming arc, there may be many adjacent NFIBSs to consider—expressed algorithmically as a variety of possible flow-change/leaving-variable/leaving-variable-status combinations, each having a different effect on the infeasibility metrics.

The four IPAs presented for the UIFTP may be summarized as follows.

IPA-1: A quick-exchange algorithm that seeks an IFIBS by starting from a given NFIBS, and constructing a series of solutions with strictly improving $f(B)$. Highly restrictive pivoting rules consider only a limited number of exchange possibilities in order to streamline the search process. Termination occurs when an IFIBS is identified or when no improving nonbasics exist.

IPA-2: A prioritized tabu-search algorithm for constructing an IFIBS from an initial NFIBS. It explores a series of solutions created through moves that consider all three improvement metrics ($\overline{c}_{ij}$, $f(B)$, and $d(B)$), combined using user-supplied weights. Consecutive IFB's need not strictly improve, and cycling is avoided with short-term memory constructs. Termination occurs when an IFIBS is identified or when no improving nonbasics exist.

IPA-3: An improvement algorithm that seeks to improve upon an existing IFIBS via strategic oscillation, which permits IF-infeasibility for diversification purposes [11], alternating with the application of IPA-1 and IPA-2.

IPA-4: A consolidated framework, combining the above IPAs into a general-purpose algorithm for (P).

2.1 Initial NFIBS Construction

Algorithms IPA-1 and IPA-2 require an initial NFIBS. Since any basic feasible solution to the embedded transportation problem will suffice (per definitions 1 and 2), an obvious choice is $\{\overline{B}, \overline{L}, \overline{\Lambda}\}$—an optimal solution to $(\overline{P})$, the linear relaxation of (P), given by (1)–(4).

2.2 IPA-1: Quick-Exchange Algorithm

Algorithm IPA-1 (given in Figure 3) evaluates nonbasics for entry into the current NFIBS using two criteria. An IF-nonbasic $(p, q) \in L \cup \Lambda$ is considered eligible for pivoting into NFIBS basis B if, after pivoting with flow change δ to form the new basis B',

1. (p, q) remains IF-feasible, and
2. the number of infeasibilities on its BEP decreases [i.e., $f(B') < f(B)$].

This narrow definition permits eligibility to be determined from a single trace of the BEP, during which the flow change and leaving arc are also identified. IF-nonbasics are placed in a circular list, from which arcs are drawn, evaluated, and, if eligible, pivoted into the current NFIBS, until no pivot-eligible arcs remain. Cycling is avoided by criterion 2, and some arcs can be eliminated from consideration by use of the theorems below. [In this discussion, an *increasing* (*decreasing*) arc is one whose flow increases (decreases) when a given pivot is performed.]

Theorem 1 *For arc $(p, q) \in L$, there is no flow change that meets the criteria if there exists a decreasing-flow IF-infeasible arc in its BEP.*

Proof: Suppose there exists an arc $(i, j) \in A^-(p, q) \cap F^c$. For (p, q) with $x_{pq} = 0$ to maintain IF-feasibility (criterion 1), $\delta \geq \ell$ or $\delta = 0$. Suppose $\delta \neq 0$. That means, $\delta \geq \ell$. But we know that $0 < x_{ij} < \ell \Rightarrow -\delta < x_{ij} < \ell - \delta$. Since $\ell - \delta < 0$, this implies that $x_{ij} < 0$. This is a contradiction to the fact that x_{ij} is nonnegative. Therefore the assumption that $\delta \neq 0$ is incorrect and $\delta = 0$ so that no flow change can occur. ∎

Theorem 2 *For arc $(p, q) \in L$, there is no flow change that meets the criteria unless there exists an increasing IF-infeasible arc in its BEP.*

Proof: By the hypothesis, there exists no arc (i, j) such that $(i, j) \in A^+(p, q) \cap F^c$. This implies that $f(B_{bep}(p, q)) = 0$. Therefore, the only infeasible arc in $B_{bep}(p, q)$ may be decreasing. By Theorem 1, no flow change occurs if there exists a decreasing infeasible arc on the BEP of (p, q). So no flow change will occur unless there exists an increasing IF-infeasible arc in $B_{bep}(p, q)$. ∎

procedure IPA-1($it, (P),\ \mathbf{x},\ B,\ L,\ \Lambda$)

inputs: $it, (P),\ \mathbf{x},\ B,\ L$[a]
outputs: $it, \mathbf{x},\ L,\ \Lambda,\ B$

begin[b]
  do while ($f(A) > 0$ and $(C_1 = \{(i,j) \in B^c : \text{IFratio}(i,j,B,\mathbf{x},\ell) > 0\}) \neq \emptyset$)
    select $(p,q) \in C_1$
    $\delta \leftarrow \text{IFratio}(p,q,B,\mathbf{x},\ell)$
    if ($(p,q) \in L$ and $\delta = \ell$) **then**
      $L \leftarrow L - (p,q)$
      $\Lambda \leftarrow \Lambda \cup (p,q)$
      update $B_{bep}(p,q)$ flows to reflect (p,q) status change
    else
      $(r,s) \leftarrow \text{argArc}(\delta)$
      call pivot$(p,q,r,s,\delta,B,B^c,\mathbf{x})$
      $it = it + 1$
    endif
  enddo
end

[a] where $\{B, L\}$ is an NFIBS.

[b] where argArc(δ_s) is the arc in $B_{bep}(p,q)$ that determined the value of δ_s, pivot$(i,j,k,l,\delta,B,B^c,\mathbf{x})$ is a procedure that performs a simplex pivot with flow change δ to bring (i,j) into and remove (k,l) from the current NFIBS

Figure 3: Procedure IPA-1

function IFratio($p,\ q,\ B,\ \mathbf{x},\ L,\ \Lambda,\ \ell$)

inputs: $p,\ q,\ \mathbf{x},\ B,\ L,\ \Lambda,\ \ell$

begin

if$((p,q) \in L)$**then**

compute[a]

$\kappa_1 \equiv \mid A^-(p,q) \cap F \cap B_1 \mid$

$\kappa_2 \equiv \mid A^+(p,q) \cap F^c \mid$

$\kappa_3 \equiv \mid A^-(p,q) \cap F^c \mid$

$\delta_5 \equiv \min_{(i,j) \in A^-(p,q) \cap F\ \cap B_1} \{x_{ij}\}$

$\delta_6 \equiv \min_{(i,j) \in A^-(p,q) \cap F\ \cap B_2} \{x_{ij}\}$

$$\delta \leftarrow \begin{cases} \delta_5, & \text{if } \kappa_1 > 0,\ \kappa_2 > \kappa_1,\ \kappa_3 = 0 \text{ and } (\delta_6 - \delta_5) \geq \ell \\ \ell, & \text{if } \kappa_1 = 0, \kappa_2 > 0, \text{ and } \kappa_3 = 0 \\ 0, & \text{otherwise} \end{cases}$$

elseif$((p,q) \in \Lambda)$**then**

compute[b]

$\kappa_2 \equiv \mid A^+(p,q) \cap F^c \mid$

$\kappa_3 \equiv \mid A^-(p,q) \cap F^c \mid$

$\delta_2 \equiv \max_{(i,j) \in A^+(p,q) \cap F^c} \{\ell - x_{ij}\}$

$\delta_3 \equiv \min_{(i,j) \in A^-(p,q) \cap F^c} \{x_{ij}\}$

$\delta_4 \equiv \min_{(i,j) \in A^-(p,q) \cap F} \{x_{ij} - \ell\}$

$$\delta \leftarrow \begin{cases} \delta_2, & \text{if } \kappa_2 > 0,\ \kappa_3 = 0, \text{ and } \delta_4 \geq \delta_2 \\ \delta_3, & \text{if } \kappa_3 > 0 \text{ and either:} \\ & \quad \kappa_2 = 0 \text{ and } \delta_4 \geq \delta_3, \text{ or} \\ & \quad \kappa_2 > 0 \text{ and } \delta_4 = \xi \\ 0, & \text{otherwise} \end{cases}$$

endif

return δ

end

[a] where B_1, B_2 are the sets of IF-basic arcs whose flow is in the range $[\ell, 2\ell)$ and $[2\ell, \xi)$, respectively, if $A^-(p,q) \cap F \cap B_1 = \emptyset$ then $\delta_5 = \xi$, and if $A^-(p,q) \cap F \cap B_2 = \emptyset$, then $\delta_6 = \xi$

[b] where if $A^+(p,q) \cap F^c = \emptyset$, then $\delta_2 = \ell$; if $A^-(p,q) \cap F^c = \emptyset$, then $\delta_3 = \ell$; and if $A^-(p,q) \cap F = \emptyset$, then $\delta_4 = \xi$

Figure 4: Function IFratio

The tabu-search methodology involves a series of "solutions" to a given problem that are created by a series of "moves." Within the interval-pivoting algorithms, a solution is an NFIBS and a *move* is a transition from one basis to another or a change in L - Λ set membership by a nonbasic arc. A move can also be thought of as a mapping from one extreme point to another, based on certain criteria.

Unlike the network simplex method which, for a given incoming arc, has a unique flow change and (in the absence of degeneracy) leaving variable, the interval-pivoting algorithms must consider a variety of flow-change and leaving-variable possibilities for nonbasics in L or Λ. Each such pivot possibility corresponds to a potential move to a different convex interval-flow subregion that may or may not be within the feasible region of the network constraints. Not all possible moves are considered by each IPA. The subset included depends on the goals of the algorithm and the attractiveness of each move for achieving these goals.

The goal of IPA-1 is the rapid identification of an IFIBS in the neighborhood of the initial NFIBS. Hence move selection was based on processing simplicity and the likelihood of reducing the number of IF-infeasible arcs. Certain κ-δ combinations were therefore considered attractive and included, while others were deemed less so and omitted. For example, one excluded move was to decrease flow on a member of Λ, which would maintain that arc's IF-feasibility only if the flow change could be exactly equal to ℓ, a move that would, at the same time, cause infeasibility on decreasing basic arcs with flows in the interval $(\ell, 2\ell)$. This restricted move set ensures that $f(A)$, the number of IF-infeasible arcs, strictly decreases with each pivot. Hence, cycling is automatically avoided and a local optimum will be reached, but many potential routes to feasibility are not explored.

2.3 IPA-2: Priority Search

IPA-2 (given in Figures 5–7) is a more flexible solution-improvement procedure that can be adapted for various goals. It broadens the range of moves that are considered, uses multiple criteria to evaluate moves, and incorporates tabu search's [8] short-term memory.

Unlike IPA-1, swapping infeasibilities (causing IF-infeasibility on some arcs to eliminate it on others) is allowed, and strict improvement in $f(A)$ is not required. Potential moves are evaluated by a weighted sum of improvement in $f(B)$, improvement in $d(B)$, and the reduced costs with respect to the current NFIBS. Priorities are reflected in the weights chosen for the three evaluation metrics. (The selection of weights is discussed in section 2.5.)

To evaluate the large number of possible moves for each nonbasic, the basis equivalent path must be traversed multiple times. This is the result of needing to know minimum and maximum flow values, and the number of basic arcs with flow in certain ranges prior to assessing possible flow changes and their respective

procedure IPA-2$((P),\ \mathbf{x},\ B,\ L,\ \Lambda,\ \ell,\ \mathbf{w})$

inputs: $(P),\ \mathbf{x},\ B,\ L,\ \ell,\ \mathbf{w}$
outputs: $\mathbf{x},\ L,\ \Lambda,\ B$

begin
  do while $(f(A) > 0$ and
  $(C_2 = \{(i,j) \in L \cup \Lambda : \text{IFpricer}(i,j,B,\mathbf{x},\ L,\ \Lambda,\ \ell)) > 0$ and $t(i,j) \geq \tau \neq \emptyset)^a$
    select $(p,q) \in C_2$
    $\delta \leftarrow \text{IFpricer}(p,q,B,\mathbf{x},\ L,\ \Lambda,\ \ell)$
    if($(p,q) \in L$ and $\delta = \ell$) **then**
      $L \leftarrow L - (p,q)$
      $\Lambda \leftarrow \Lambda \cup (p,q)$
      update $B_{bep}(p,q)$ flows to reflect (p,q) status change
    else
      $(r,s) \leftarrow \text{argArc}(\delta)$
      call pivot$(p,q,r,s,\delta,B,B^c,\mathbf{x})$
    endif
  enddo
end

[a] where $t(i,j)$ is the number of immediately preceding, consecutive iterations that arc (i,j) has been IF-nonbasic.

Figure 5: Procedure IPA-2

function IFpricer($p,\ q,\ B,\ \mathbf{x},\ L,\ \Lambda,\ \ell$)

inputs: $p,\ q,\ B,\ \mathbf{x},\ L,\ \Lambda,\ \ell$

begin
  $\Delta \leftarrow \text{pfs}(i,\ j,\ \mathbf{w},\ B,\ \mathbf{x},\ \ell)$
  $\delta_{best} \leftarrow \delta_0 \in \Delta$ such that $\hat{c}(i,j,\delta_0,\mathbf{w}) = \max_{\delta_s \in \Delta}\{\hat{c}(i,j,\delta_s,\mathbf{w}), 0\}$
  return δ_{best}
end[a]

---

[a] where

$$
\begin{aligned}
w_1 &\equiv \text{weight for number-of-infeasibilities metric} \\
w_2 &\equiv \text{weight for distance-of-feasibilities metric} \\
w_3 &\equiv \text{weight for the reduced cost} \\
\mathbf{w} &\equiv (w_1, w_2, w_3) \\
g(i,j,\delta) &\equiv f(B) - f(B - argArc(\delta) \cup (i,j)) \\
h(i,j,\delta) &\equiv d(B) - d(B - argArc(\delta) \cup (i,j)) \\
\hat{c}(i,j,\delta,\mathbf{w}) &\equiv \mathbf{w} \cdot (g(i,j,\delta), h(i,j,\delta), \bar{c}_{ij})
\end{aligned}
$$

Figure 6: Function IFpricer

function pfs($p,\ q,\ B,\ \mathbf{x},\ \ell,\ \Delta$)

inputs: $p,\ q,\ B,\ \mathbf{x},\ \ell,\ \Delta$

begin

compute

$\kappa_2(p,q) \equiv \mid A^+(p,q) \cap F^c \mid$

$\kappa_3(p,q) \equiv \mid A^-(p,q) \cap F^c \mid$

$\delta_1(p,q) \equiv \min_{(i,j)\in A^+(p,q)\cap F^c} \{\ell - x_{ij}\}$

$\delta_2(p,q) \equiv \max_{(i,j)\in A^+(p,q)\cap F^c} \{\ell - x_{ij}\}$

$\delta_3(p,q) \equiv \min_{(i,j)\in A^-(p,q)\cap F^c} \{\ell - x_{ij}\}$

$\delta_4(p,q) \equiv \min_{(i,j)\in A^-(p,q)\cap F} \{x_{ij} - \ell\}$

$\delta_5(p,q) \equiv \min_{(i,j)\in A^-(p,q)\cap F\ \cap B_1} \{x_{ij}\}$

$$\Delta \leftarrow \begin{cases} \{\delta_1,\delta_2\}, & \text{if } \kappa_3(p,q)>0,\ \kappa_2(p,q)>0, \text{ and } \delta_3(p,q)>\delta_2(p,q) \\ \{\delta_1\}, & \text{if } \kappa_3(p,q)>0,\ \kappa_2(p,q)>0, \text{ and } \delta_3(p,q)\le\delta_2(p,q) \\ \{\delta_3,\delta_4\}, & \text{if } \kappa_3(p,q)>0,\ \kappa_2(p,q)=0, \text{ and } \delta_3(p,q)>\delta_4(p,q) \\ \{\delta_3\}, & \text{if } \kappa_3(p,q)>0,\ \kappa_2(p,q)=0, \text{ and } \delta_3(p,q)\le\delta_4(p,q) \\ \{\delta_1,\delta_2,\delta_4,\delta_5\}, & \text{if } \kappa_3(p,q)=0,\ \kappa_2(p,q)>0 \text{ and } \delta_3<2\ell \\ \{\delta_1,\delta_2,\delta_3,\delta_4,\delta_5\}, & \text{if } \kappa_3(p,q)=0,\ \kappa_2(p,q)>0 \text{ and } \delta_3\ge 2\ell \\ \emptyset, & \text{otherwise} \end{cases}$$

return Δ

end

Figure 7: Function pfs

effects on the evaluation criteria. These move possibilities are inherent in the κ-δ combinations computed and evaluated in Figure 7.

Short-term memory is instituted through the concept of the *tenure*, $t(i,j)$, of a nonbasic arc (i,j), defined as the number of continuous previous iterations that the arc has remained nonbasic.[1] Arcs are restricted from basis entry if their tenure does not exceed a user-supplied parameter, τ. When an arc leaves the basis, its tabu status is set and it must remain nonbasic for at least τ iterations. This tabu status helps avoid cycling and drives the search away from recently explored neighborhoods.

Although not shown in the pseudo-code, it is helpful to employ a *candidate list* mechanism [14] when implementing IPA-2. In this heuristic, an entering arc is chosen from a list of nonbasics that are deemed attractive by the designated strategy. The list of candidates can be initiated by scanning nonbasics in blocks of m_2 arcs, and retaining the m_1 best. For each iteration of IPA-2, all candidate arcs are examined using the move evaluation criteria and the best of the attractive candidates, if any, is selected for pivoting and removed from the list. Since the metrics for each nonbasic may change after each move, the remaining candidates must be re-evaluated. The selection process is repeated until no attractive candidate remains, at which point the list is refreshed using the (m_1, m_2) rule.

2.4 IPA-3: Improvement Search

The IPA-3 *improvement search* begins with an initial IFIBS and applies *strategic oscillation* [11, 7, 8] diversification to move away from IF-feasibility temporarily in hopes of returning (via IPA-1 and IPA-2) to another region of the feasibility space that contains a better IFIBS.[2] The primary idea is to allow feasible and infeasible solutions to occur, so that the search is able to explore more of the solution space and locate a better solution in the process. Since the feasible space for (P) is disjoint, the strategic oscillation provides a diversification mechanism that crosses a region of infeasibility in search of a better solution. Termination of the process is controlled, in part, by *IPA-gap*, the difference between the objective function value of the initial relaxed problem, z^*_P, and the objective function value of the best-found IFIBS incumbent, z_{inc}, expressed as a percentage: $\frac{(z_{inc} - z^*_P)}{z^*_P} \cdot 100$. This is the farthest the incumbent is from the optimal solution. The gap is compared with the user-supplied parameter ϵ.

[1] Tenure can be easily maintained by initializing the start of tenure at 0, updating the start when an arc leaves the basis, and computing $t(i,j)$ from the current iteration number as needed.

[2] There are a variety of longer term memory strategies in tabu search, such as those based on frequency memory. We restrict ourselves to a particular variant of strategic oscillation (see, e.g., sections 4.2 and 4.7 of [8]) that is straightforward, for the purpose of determining the basic merit of our approach without resorting to a sophisticated guidance process at the metaheuristic level.

procedure IPA-3($it, (P),\ \mathbf{x},\ B,\ L,\ \Lambda,\ \mathbf{w},\ \epsilon,\ \theta,\ \alpha,\ \eta,\ \gamma$, *IPA-Gap*)

inputs: $it, (P),\ \mathbf{x},\ B,\ L,\ \Lambda,\ \mathbf{w},\ \epsilon,\ \theta,\ \alpha,\ \eta$, *IPA-Gap*
outputs: $it, \mathbf{x},\ L,\ \Lambda,\ B$
begin
  initialize att = 0, citr = 0
  do while (*IPA-Gap* $> \epsilon$ and att $< \eta$ and citr $< \gamma$) [a]
    $f(B) \leftarrow$ diverse$(it, (P),\ \mathbf{x},\ B,\ L,\ \Lambda,\ \theta,\ \alpha)$
    call IPA-1$((P),\ \mathbf{x},\ B,\ L,\ \Lambda)$
    if $(F^c \neq \emptyset)$ **call** IPA-2$((P),\ \mathbf{x},\ B,\ L,\ \Lambda,\ \mathbf{w})$
    if $(F^c \neq \emptyset)$ **then**
      att = att + 1
    else
      update *IPA-Gap*
      if (*IPA-Gap* $> \epsilon$) citr = citr + 1
    endif
  enddo
end

[a] where *IPA-Gap* is the maximum optimality gap, *att* is the number of unsuccessful attempts to find a feasible solution to (P), and *citr* is the number of consecutive non-improving iterations

Figure 8: Procedure IPA-3: Improvement Search

Improvement search works as follows: Using a special ratio test, the problem is driven into an infeasible region. Then, IPA-1 and IPA-2 are applied to move back into a feasible region. If this succeeds, then the objective function value is computed. If this is better than z_{inc}, it is saved. This process is repeated using a parameter that can be adjusted. The procedure terminates when either: (1) the *IPA-gap* is smaller than ϵ, (2) no improvement has been made in η consecutive iterations, or (3) γ iterations have been performed. The IPA-3 algorithm is defined in Figure 8.

2.5 IPA-4: Consolidated Algorithm for UIFTP

The IPA-4 procedure is given in Figure 10. An initial IFIBS is sought by first optimizing $(\overline{P})$ then, if necessary, applying IPA-1 and possibly IPA-2. (If this process fails, the algorithm terminates without finding an IF-feasible solution.[3])

[3] This condition did not occur in any of our tests.

function diverse $(it, (P),\ \mathbf{x},\ B,\ L,\ \Lambda,\ \theta,\ \alpha)$
inputs: $it, (P),\ \mathbf{x},\ B,\ L,\ \Lambda,\ \theta,\ \alpha$

begin
  initialize itt $= 0$
  do while (itt $< \alpha$ and $f(B) < \theta$) [a]
    select $(p, q) \in B^c$
    call simpivot$(p, q, B, B^c, \mathbf{x})$
    update $B, B^c, \mathbf{x}$
    compute $f(B)$
    $it = it + 1$
    itt = itt + 1
  enddo
  return $f(B)$
end

[a] where simpivot$(p,\ q, B, B^c, \mathbf{x})$ performs a network simplex pivot to bring $(p, q) \in B^c$ into the current basic solution B, and itt is the number of pivots performed.

Figure 9: Procedure diverse

IPA-3 is then applied to improve on the initial solution.

The modified ratio test is based on IF-feasibility. The pricing procedure is based on the metrics, $f(B)$, $d(i, j)$, and $\bar{c}_{ij}$, as discussed earlier. The number of infeasibilities affected by the incoming nonbasic is measured by traversing the basis equivalent path (BEP) of this nonbasic. The change in distance from feasibility due to the possible incoming nonbasic is also computed by traversing the BEP of this nonbasic, computing the possible flow change. Lastly, the reduced cost is evaluated to ensure that the best solution on that hyperplane is obtained.

The following strategy was employed in our use of IPA-4 and choice of associated parameters. For IPA-2, the initial NFIBS may not be an IFIBS, in which case priority is given to driving the problem closer to feasibility. Since the application of IPA-2 immediately follows IPA-1, whose single criterion is reducing $f(A)$, preference should be given to the distance-from-feasibility metric, then to the reduced cost. By reflecting this relationship in the weights, $\mathbf{w}$, feasibility is the overriding choice factor and the reduced costs break ties and lead to the best solution on the selected hyperplane. The short-term memory works well in avoiding cycling.

procedure IPA-4$(it, (P),\ \mathbf{w},\ \epsilon,\ \theta,\ \alpha,\ \eta,\ \beta,\ \mathbf{x},\ L,\ \Lambda,\ B)$
inputs: $it, (P),\ \mathbf{w},\ \epsilon,\ \theta,\ \alpha,\ \eta,\ \beta$
outputs: $it, \mathbf{x},\ L,\ \Lambda,\ B$

begin
  initialize $it = 0$
  do while ($it < \beta$)[a]
    call purenet$(it, (\overline{P}), \mathbf{x}, L)$[b]
    if ($F^c = \emptyset$) **then**
      report optimal solution
    else
      call IPA-1$(it, (P), \mathbf{x}, B, L, \Lambda)$
      if ($F^c \neq \emptyset$) **call** IPA-2$(it, (P),\ \mathbf{x},\ B,\ L,\ \Lambda,\ \mathbf{w})$
      if ($F^c \neq \emptyset$) **then**
        Stop; (P) may be infeasible
      else
        compute *IPA-Gap*
        if (*IPA-Gap* $> \epsilon$) **call** IPA-3$(it, (P),\ \mathbf{x},\ B,\ L,\ \Lambda,\ \mathbf{w},$
         $\epsilon,\ \theta,\ \alpha,\ \eta,\ \gamma,$ *IPA-Gap*)
      endif
    endif
  enddo
end

[a] where it is the total number of pivots performed in purenet( ), IPA-1( ), IPA-2( ), and IPA-3( ), combined.

[b] where purenet$(it, (\overline{P}), \mathbf{x}, L)$ is a procedure that optimizes $(\overline{P})$ and updates $it, \mathbf{x}, L$.

Figure 10: Procedure IPA-4

3 Experimental Design

To test the efficacy of the IPA approach, an empirical experiment was undertaken to (1) compare an IPA-4 implementation with a commercial-grade software system and (2) identify factors that affected performance. Details of the testing and an analysis of the results are presented below.

3.1 Software Tested

To assess the effectiveness of the interval-pivoting algorithms, all were implemented in FORTRAN-77, and the IPA-4 consolidated algorithm was used for benchmarking. This IPA-NET software uses the data structures of [2] and the NETFLO2 [18] routines for solving the initial network relaxation and executing pivots. IPA-NET was compared with CPLEX, Version 4.0, an exact branch-and-cut code for identifying optimal and ϵ-optimal solutions.

Hence, this testing compares an exact, general-purpose optimizer with a special-purpose heuristic-based code, giving IPA-NET a natural advantage. However, the comparison highlights any practical benefits from the use of either software package and provides insight into the solvability of uniform interval-flow transportation problems. CPLEX was run with a limit of 20,000 subproblems and default values for all other parameters. For IPA-NET, a modest amount of preliminary testing on a small, diverse group of problems yielded settings for the heuristic's parameters that appeared to be robust and were used in all reported results. These settings are: $\beta = 10000, \alpha = 100, \gamma = 5, \tau = 2, m_1 = 20, m_2 = 7, \mathbf{w} = (0, 100000, 1)$, and $\theta = f(\overline{B}) + 1$. Both codes were run with $\epsilon = 10^{-4}$.

3.2 Test Problems

Because interval-flow networks are a new problem class, no standard test sets of UIFTP instances were available. This experiment used 69 randomly generated problem instances created by the well-known, machine-independent NETGEN program [12], modified to compute a uniform conditional lower bound for each. NETGEN permits the designation of a variety of instance characteristics, thus facilitating the creation of test sets to support an experimental design.

As determined by Property 5, $b_{min} = \min_{i \in N}\{|b_i|\}$ is an upper bound on ℓ for problem feasibility. Since NETGEN does not provide direct control of the range nor distribution of the b_i, b_{min} was determined a posteriori and ℓ computed as a user-defined percentage of $(\%b_{min})$. To maintain integrality of $\ell = \lfloor (\%b_{min})(b_{min})/100 \rfloor$, was used. Generated problems with $b_{min} \leq 7$ were not included in the testing because of the triviality of the resulting ℓs. Two test sets were created for the experiment. Test set A is made up of smaller instances that were expected to be solvable by both IPA-NET and CPLEX. Instances in this set have from 50 to 250 nodes and from 100 to 2000 arcs. Set B consisted

of larger problems, considered to be beyond the ready solvability of CPLEX on our machine, with up to 5000 nodes and 10,000 arcs.

Test set A was designed not only to compare the two codes, but to permit statistical analysis of any effects of problem characteristics on problem run-times. The factors and levels used in the creation of test set A are given in Table 1. The 64 problems in set A are described in Tables 2–5 and include an instance of each combination of the six factors described in Table 1. The largest problems had 100 network nodes, 2100 constraints, 2000 continuous variables, and 2000 binary variables.

Test set B consisted of five larger problems, with 1000 to 5000 nodes, 2000 to 10,000 arcs (binary and continuous variables), 60% source nodes, a maximum cost of 1000, an average supply of 2500 units, and a conditional lower bound of 75% of b_{min}. All of the NETGEN parameters needed to create these problems are detailed in Table 6.[4]

3.3 Performance Measures

Performance of the two solution approaches is evaluated in terms of solution time and solution quality. The times used are total program execution time, excluding problem input and output of the results. Also reported are the number of pivots (iterations) performed.

Solution quality is measured in two ways. First, the difference in values between the IPA-NET heuristic's best solution value and the CPLEX ϵ-optimal value is expressed as a percentage of the CPLEX value. Secondly, the IPA gap measures the heuristic's closeness to optimality, based exclusively on internally derived values: the difference between z_{inc}, the value of the best IFIBS, and $z^*_{\overline{P}}$, the value of the optimal solution to $(\overline{P})$, expressed as a percentage of $z^*_{\overline{P}}$. Benefit/cost analyses consider a ratio of the percent of optimality to total execution time.

4 Empirical Results and Statistical Analyses

All computer runs were made at Southern Methodist University on a DEC AlphaServer 2100 with a 5/250 megahertz EV-5 processor, four megabytes of b-cache, 256 megabytes of random-access memory, and running under the OSF/1 3.2g Unix operating system. All testing was performed on a lightly loaded system under similar job mixes. IPA-NET was compiled with the DEC `f77` compiler using the default optimization options. Reported IPA-NET times were user execution times provided by the `clock()` system function.

[4]The modified NETGEN code and test problems are available from the author.

Table 1: Test Set A Problem Factors and Levels

Factor	Level 1	Level 2
Number of nodes	50	100
Number of arcs (nested within nodes)		
In 50-node problems	250	500
In 100-node problems	1,000	2,000
Percent source nodes	50%	60%
Maximum cost	100	10,000
Average units of supply	250	2500
ℓ as % of b_{min}	25%	75%

Table 2: Test Set A: Problems 1–16 (50 nodes, 250 arcs)

Prob. No.	Random Seed	Total Nodes	Supply Nodes	Demand Nodes	Total Arcs	Max. Cost	Total Supply	$\%b_{min}$
1	5375842	50	25	25	250	100	6250	25
2	7524825	50	30	20	250	100	7500	25
3	7463159	50	25	25	250	100	6250	75
4	2378495	50	30	20	250	100	7500	75
5	6304325	50	25	25	250	100	62500	25
6	4570849	50	30	20	250	100	75000	25
7	3574609	50	25	25	250	100	62500	75
8	2954567	50	30	20	250	100	75000	75
9	4234567	50	25	25	250	10000	6250	25
10	8545642	50	30	20	250	10000	7500	25
11	7342849	50	25	25	250	10000	6250	75
12	8714359	50	30	20	250	10000	7500	75
13	7374809	50	25	25	250	10000	62500	25
14	7314809	50	30	20	250	10000	75000	25
15	2352859	50	25	25	250	10000	62500	75
16	4234567	50	30	20	250	10000	75000	75

Table 3: Test Set A: Problems 17–32 (50 nodes, 500 arcs)

Prob. No.	Random Seed	Total Nodes	Supply Nodes	Demand Nodes	Total Arcs	Max. Cost	Total Supply	$\%b_{min}$
17	2572438	50	25	25	500	100	6250	25
18	4934567	50	30	20	500	100	7500	25
19	1530567	50	25	25	500	100	6250	75
20	2314359	50	30	20	500	100	7500	75
21	9714359	50	25	25	500	100	62500	25
22	3310359	50	30	20	500	100	75000	25
23	4314359	50	25	25	500	100	62500	75
24	5273439	50	30	20	500	100	75000	75
25	5463759	50	25	25	500	10000	6250	25
26	9463159	50	30	20	500	10000	7500	25
27	3415129	50	25	25	500	10000	6250	75
28	7533759	50	30	20	500	10000	7500	75
29	4454567	50	25	25	500	10000	62500	25
30	4474325	50	30	20	500	10000	75000	25
31	88463259	50	25	25	500	10000	62500	75
32	1934567	50	30	20	500	10000	75000	75

Table 4: Test Set A: Problems 33–48 (100 nodes, 1000 arcs)

Prob. No.	Random Seed	Total Nodes	Supply Nodes	Demand Nodes	Total Arcs	Max. Cost	Total Supply	$\%b_{min}$
33	1445642	100	50	50	1000	100	12500	25
34	4954567	100	60	40	1000	100	15000	25
35	1374809	100	50	50	1000	100	12500	75
36	7423474	100	60	40	1000	100	15000	75
37	4375842	100	50	50	1000	100	125000	25
38	6424025	100	60	40	1000	100	150000	25
39	8472649	100	50	50	1000	100	125000	75
40	4574849	100	60	40	1000	100	150000	75
41	1374809	100	50	50	1000	10000	12500	25
42	4574609	100	60	40	1000	10000	15000	25
43	1375849	100	50	50	1000	10000	12500	75
44	4025419	100	60	40	1000	10000	15000	75
45	5674609	100	50	50	1000	10000	125000	25
46	1570849	100	60	40	1000	10000	150000	25
47	5273439	100	50	50	1000	10000	125000	75
48	1534567	100	60	40	1000	10000	150000	75

Table 5: Test Set A: Problems 49 - 64 (100 nodes, 2000 arcs)

Prob. No.	Random Seed	Total Nodes	Supply Nodes	Demand Nodes	Total Arcs	Max. Cost	Total Supply	$\%b_{min}$
49	1375805	100	50	50	2000	100	12500	25
50	5272438	100	60	40	2000	100	15000	25
51	1573408	100	50	50	2000	100	12500	75
52	1473504	100	60	40	2000	100	15000	75
53	1573504	100	50	50	2000	100	125000	25
54	2573458	100	60	40	2000	100	150000	25
55	1673504	100	50	50	2000	100	125000	75
56	8573604	100	60	40	2000	100	150000	75
57	1374829	100	50	50	2000	10000	12500	25
58	7714359	100	60	40	2000	10000	15000	25
59	4570829	100	50	50	2000	10000	12500	75
60	7530567	100	60	40	2000	10000	15000	75
61	1340507	100	50	50	2000	10000	125000	25
62	1454567	100	60	40	2000	10000	150000	25
63	2034560	100	50	50	2000	10000	125000	75
64	1375495	100	60	40	2000	10000	150000	75

Table 6: Test Set B

Prob. No.	Random Seed	Total Nodes	Supply Nodes	Demand Nodes	Total Arcs	Max. Cost	Total Supply	$\%b_{min}$
1	1229267	1000	600	400	2000	1000	1500000	75
2	1559267	2000	1200	800	4000	1000	3000000	75
3	1259267	3000	1800	1200	6000	1000	4500000	75
4	4449267	4000	2400	1600	8000	1000	6000000	75
5	7759267	5000	3000	2000	10000	1000	7500000	75

Table 7: Empirical Results, Test Set A : 50 nodes, 250 arcs

Prob	Final Objective Value[a]				Time, (Seconds)		IPA
#	IPA-NET	CPLEX		z_{diff}	IPA-NET	CPLEX	Gap
1	104835	**104287**		0.52	**0.03**	1.73	0.54
2	**159467**	**159467**		0.00	**0.00**	1.83	0.00
3	1402550	**1397310**		0.37	**0.03**	1.02	0.49
4	169331	**168230**		0.65	**0.03**	0.37	0.69
5	**1451459**	**1451459**		0.00	**0.00**	1.30	0.00
6	1726064	**1724055**		0.11	**0.03**	1.75	0.16
7	1503007	**1502514**		0.03	**0.03**	1.92	0.06
8	1701889	**1701430**		0.02	**0.03**	4.18	0.04
9	**17370455**	**17370455**		0.00	**0.00**	0.80	0.00
10	16731250	**16560583**		1.03	**0.00**	4.72	1.19
11	11334279	**10953984**		3.47	**0.00**	8.12	4.02
12	17924424	**17714746**	*	1.18	**0.01**	126.95	1.64
13	**158940430**	**158940430**		0.00	**0.00**	0.60	0.00
14	**126367221**	**126367221**		0.00	**0.00**	0.27	0.00
15	**131554052**	131611472	*	-0.04	**0.03**	104.95	0.55
16	141900938	**140475167**		1.01	**0.00**	24.72	1.26

[a]where * indicates the instance for which CPLEX terminated at 20,000-subproblem limit

4.1 Test Set A: Computational Results and Analysis

Tables 7–10 give the codes' observed performance metrics for the 64 problem instances in Test Set A. Shown for each instance is the objective function value of the best solutions found by each code, the difference in solution values as a percentage of the CPLEX value (z_{diff}), the *IPA-gap*, and the execution times for each code. The better solution values and times are highlighted with boldface. Those instances for which CPLEX terminated at the 20,000-subproblem limit are indicated by an asterisk (*); all other CPLEX values were for solutions known to be within 0.01% of optimality. These results are summarized in Table 11, which groups the observations by number of nodes, number of arcs, and size of the conditional lower bound.

4.1.1 Solution Quality

Both codes found feasible solutions for all problems. In three cases, IPA-NET found a solution that was superior to CPLEX's, in 39 instances CPLEX identified the superior solution, and in 19 cases the solution quality was equivalent.

Table 8: Empirical Results, Test Set A : 50 nodes, 500 arcs

Prob	Final Objective Value[a]				Time, (Seconds)		IPA
#	IPA-NET	CPLEX		z_{diff}	IPA-NET	CPLEX	Gap
17	**114791**	**114791**		0.00	**0.00**	1.75	0.00
18	**12865632**	**12865632**		0.00	**0.00**	2.67	0.00
19	84092	**82152**		2.36	**0.00**	2.13	2.38
20	116055	**113905**	*	1.88	**0.00**	142.98	2.23
21	**776362**	**776362**		0.00	**0.00**	2.20	0.00
22	1079541	**1079463**		0.00	**0.01**	0.57	0.02
23	712993	**710078**		0.41	**0.04**	4.73	0.54
24	969955	**956428**		1.41	**0.01**	3.42	1.52
25	**6950544**	**6950544**		0.00	**0.01**	0.67	0.00
26	9630721	**9569913**		0.63	**0.01**	2.63	0.66
27	6225013	**6006418**		3.63	**0.00**	3.35	3.65
28	8067529	**7958239**		1.37	**0.03**	56.40	1.81
29	**88862250**	**88862250**		0.00	**0.00**	2.53	0.00
30	**95348544**	**95348544**		0.00	**0.00**	3.23	0.00
31	84026784	**78343260**	*	7.25	**0.00**	207.00	8.49
32	94045378	**93342978**		0.75	**0.03**	4.20	0.79

[a]where * indicates the instance for which CPLEX terminated at 20,000-subproblem limit

Table 9: Empirical Results, Test Set A : 100 nodes, 1000 arcs

Prob	Final Objective Value[a]				Time, (Seconds)		IPA
#	IPA-NET	CPLEX		z_{diff}	IPA-NET	CPLEX	Gap
33	**208140**	**208140**		0.00	**0.01**	4.60	0.00
34	2180980	**2178825**		0.09	**0.04**	35.75	0.15
35	174468	**171358**		1.81	**0.03**	88.00	1.86
36	**241702**	**241702**		0.00	**0.03**	6.28	0.00
37	**1999598**	**1999598**		0.00	**0.03**	18.63	0.00
38	2383290	**2382476**		0.03	**0.04**	7.77	0.23
39	2116282	**2057301**		2.86	**0.01**	107.88	3.07
40	2377495	**2356055**		0.90	**0.04**	30.73	0.92
41	**20750074**	**20750074**		0.00	**0.03**	10.22	0.00
42	26891953	**26885413**		0.02	**0.03**	5.23	0.02
43	23847702	**23745647**		0.42	**0.06**	65.08	0.51
44	24906611	**24732655**		0.70	**0.04**	12.38	0.75
45	280747180	**280726970**		0.00	**0.01**	8.07	0.00
46	25484951	**25454951**		0.11	**0.01**	14.30	0.00
47	193395208	**192854033**		0.28	**0.03**	21.87	0.29
48	249363842	**244673658**	*	1.91	**0.03**	548.27	2.15

[a]where * indicates the instance for which CPLEX terminated at 20,000-subproblem limit

Table 10: Empirical Results, Test Set A : 100 nodes, 2000 arcs

Prob	Final Objective Value[a]				Time, (Seconds)		IPA
#	IPA-NET	CPLEX		z_{diff}	IPA-NET	CPLEX	Gap
49	123008	**122924**		0.06	**0.03**	14.37	0.06
50	92074	**91926**	*	0.16	**0.06**	688.18	0.18
51	133428	**132931**		0.37	**0.06**	322.50	0.53
52	**141473**	141533	*	-0.04	**0.08**	379.15	0.59
53	**1419263**	**1419263**		0.00	**0.03**	24.98	0.00
54	**1624880**	**1624880**		0.00	**0.01**	4.37	0.00
55	1178636	**1178198**		0.03	**0.11**	83.20	0.04
56	1088035	**1050060**		3.61	**0.03**	93.23	3.68
57	**11992891**	11993608		-0.00	**0.01**	25.87	0.00
58	**13153453**	**13153453**		0.00	**0.03**	22.17	0.00
59	11083279	**11023021**		0.54	**0.03**	9.08	0.56
60	11764190	**11591995**		1.48	**0.09**	87.87	1.51
61	74762401	**74658311**		0.13	**0.06**	12.42	0.15
62	**128020513**	**128020513**		0.00	**0.01**	43.22	0.00
63	90764354	**90438020**		0.36	**0.03**	11.38	0.36
64	133952213	**131158660**		2.12	**0.03**	563.88	2.21

[a]where * indicates the instance for which CPLEX terminated at 20,000-subproblem limit

Table 11: Test Set A : Summary

Nodes	Arcs	% b_{min}	Average % z_{diff}	Average Time, Seconds IPA-NET	CPLEX	Ratio[a]
50	250	25	0.20	0.00	1.62	216.00
50	250	75	0.83	0.02	34.02	1701.00
50	500	25	0.07	0.00	2.03	541.33
50	500	75	2.38	0.01	53.02	5302.00
100	1000	25	0.02	0.02	13.07	653.50
100	1000	75	1.10	0.02	110.06	5503.00
100	2000	25	0.04	0.03	104.44	3481.33
100	2000	75	1.05	0.05	193.78	3875.60
Grand Average			0.71	0.02	64.00	2659.17

[a]Ratio of the total CPLEX time to total IPA time for all eight problems in this group

Table 12: Summary of Iterations for Test Set A

Nodes	Arcs	% b_{min}	Average Iterations IPA-NET	CPLEX	Iterations per Second IPA-NET	CPLEX
50	250	25	143.00	1497.50	19066.66	924.38
50	250	75	147.67	24822.87	7383.50	729.65
50	500	25	133.99	10724.12	$\rightarrow \infty$	5282.81
50	500	75	126.37	12913.75	12637.50	243.56
100	1000	25	242.62	8466.75	12131.00	647.80
100	1000	75	237.57	22293.37	11878.50	202.55
100	2000	25	238.99	23020.12	7966.33	220.41
100	2000	75	233.75	6604.75	4675.00	34.07
Grand Average			187.99	13792.52	13933.58	1035.65

Despite this expected strong showing by the CPLEX optimizer, the quality of solutions derived by the IPA-4 heuristic was also high. As shown in Table 11, the mean z_{diff} was only 0.71%.

The solution quality is also captured in the *IPA-gap* values, which expresses the maximum optimality gap as a percentage of the initial relaxation's value. This gap in Table 14 is given as +0.00 to indicate a very small (< 0.0005) number. Also shown is z_{inc}, the objective function value of the solution found by IPA-NET.

4.1.2 Factors Affecting Solution Quality

An analysis of variance of solution value differences was performed to identify any contributing factors to the response variable z_{diff}. The factors tested were: number of nodes, number of arcs (nested within nodes), percentage of source nodes (a measure of problem rectangularity), average supply per source, maximum cost, and percentage of b_{min} used for ℓ (PCTUB).

The observations were processed by the ANOVA procedure of the SAS statistical analysis software (version 6.10). At the 5% significance level, the only factor affecting z_{diff} in a statistically significant manner was PCTUB. The Tukey's significant difference (TSD) test showed that z_{diff} was significantly higher at the PCTUB level of 75% (1.35% mean value) than at the 25% level (0.09% mean value). The other factors, including problem dimensions, did not appear to affect the quality of the IPA-NET-generated solutions.

4.1.3 Solution Time

In terms of execution speed, IPA-NET completely dominates CPLEX on Test Set A, exhibiting dramatically smaller times on all 64 problems. Recorded times ranged from 0.00 (too brief to be measured) to 0.11 seconds for the IPA-NET heuristic and from 0.27 to 688.18 seconds for CPLEX (which terminated without a proven-optimal solution in seven cases). Table 11 shows the mean solution times and time ratios by problem group and overall. The total time required for all problems was 4096.62 seconds for CPLEX and 1.53 seconds for IPA-NET (a 2677:1 ratio). Hence, on average, IPA-NET is three orders of magnitude faster than CPLEX on these problems.

The large solution time disparities are due to the heuristic and specialized nature of the IPA code. Given sufficient time, CPLEX will arrive at a guaranteed (ϵ-) optimal solution, and uses general-purpose data structures and solution techniques, not ones customized for network flow problems. The IPA-based algorithms represent constraints (3)–(5) implicitly, while CPLEX, treating the problem as a general MIP, includes them explicitly.

As a measure of the relative computational effort required for the two codes, Table 12 gives the average number of iterations performed by each for the problem groups in Test Set A. For CPLEX, the iterations are the total number of

simplex pivots executed by CPLEX in solving all subproblems, and the number of pivots executed by IPA-NET in solving the initial relaxation and applying the IPA-4 algorithm. These statistics show that an average of 84 times more iterations were required to optimize (in most cases) a problem than the heuristic needed to complete its search. The efficiency of the specialized approach is further emphasized by the "iterations per second" statistic reported in Table 12 and computed as the ratio of the average iterations to average time in seconds for each code for each problem group (shown as "$\rightarrow \infty$" for those groups with unmeasurable times). While the time portion of this metric includes all algorithmic operations, it generally reflects the number of pivots that can be selected and executed in a given machine-second. The value of network-structure exploitation and relative algorithmic simplicity is apparent from IPA-NET's ability to perform iterations an average of 13 times faster than CPLEX.

4.1.4 Factors Affecting Solution Time

An analysis of variance using execution time as the response variable was performed to identify any problem characteristics that affected solution difficulty in a statistically significant way. As before, all of the problem-generation characteristics given in Table 1, plus *computer code applied* (CODE), were used as factors. The 64 observations from Test Set A were processed by the SAS ANOVA procedure using factorial designs with number of arcs nested within number of nodes.

Based on a 5% significance level, the factors with significant effects were CODE, number of nodes (NODES), and PCTUB. There were interaction effects between CODE and NODES, and CODE and PCTUB, indicating that solution time was affected by the particular combination of code used and number of nodes, and by the combination of code used and the size of the conditional lower bound. Surprisingly, the number of problem arcs did not have a significant effect on solution time.

4.1.5 Quality-Time Tradeoff

The key justification for the use of inexact algorithms for problem-solving is in the tradeoff between solution quality and time required to achieve that quality. This type of benefit-cost analysis is particularly appropriate when the value of a solution depends on its timeliness.

From the statistical analyses it was determined that the code used and size of a problem's conditional lower bound affected both solution time and solution quality. Larger problems and problems with larger conditional lower bounds took significantly longer and yielded significantly different solution values between the two codes. This reinforces the intuition that smaller problems and problems with lower arc-usage requirements are easier to solve.

Evaluating the actual quality-time tradeoff involves a comparison of solution

quality and time required to achieve that solution. This relationship can be captured, in part, by the ratio of two values: (1) the value of the best solution found by a given code, expressed as a percentage of the best known solution value, and (2) the time in seconds to execute the code that produced that solution. This *rate-of-progress* metric can be viewed as the percentage (of optimality or best) achieved per second of run time. Table 13 shows the average value of this ratio for each code on each group of problems in Test Set A.

One interpretation of the grand means in Table 13 would be that, on average, CPLEX achieves 15% of a problem's optimal value in each second of run time. During that same second, IPA-NET can find its near-optimal solution values for 146 problems. While such averages and ratios can be misused, in this case they justifiably underscore IPA-NET as being clearly superior in a benefit-cost analysis.

4.2 Test Set B: Computational Results and Analysis

For the large-scale problems in test set B, IPA-NET maintained the high efficiency exhibited in the smaller problems (see Table 14). As indicated by the small *IPA-gap*'s (ranging from 0.000051 to 0.000447), the quality remained high, even with PCTUB fixed at its more difficult level.

The solution times ranged from 0.24 seconds for a 1000-node, 2000-arc problem to 4.23 seconds for a 5000-node, 10,000-arc interval-flow network. Given that the latter reflected a 20,000-variable, 15,000-constraint mixed-integer program, the solution times are extremely fast. Hence, the interval-pivoting code had strong performance, in terms of both solution time and solution quality for both test sets.

5 Analysis of the Results and Conclusions

Compared with state-of-the-art MIP software, IPA-NET exhibits remarkable performance in both time and solution quality. When evaluated strictly on its own merits, the heuristic terminates quickly, often with times too brief to record, and provides solutions whose values are typically within 1% of optimal. On 64 problem instances with up to 2000 integer and 2000 continuous variables, IPA-NET found solutions as good or better than CPLEX 39% of the time. For these problems, CPLEX required over an hour of processing, while IPA-NET used only 1.5 seconds.

Interval-pivoting algorithms are designed to exploit the mature and efficient optimization technologies developed for network flow problems. They can be embedded within metaheuristic solution strategies, such as tabu search and strategic oscillation. And, unlike some heuristics, they are able to determine both lower and upper bounds on the optimal solution value. Hence, this work

Table 13: Cost-Benefit Analysis for Test Set A

Nodes	Arcs	% *mlb*	% Best Known Solution/per Second IPA-NET	CPLEX
50	250	25	79840.00	61.72
50	250	75	4958.50	2.93
50	500	25	7267.63	49.26
50	500	75	9762.00	1.88
100	1000	25	4999.00	7.65
100	1000	75	4945.00	0.90
100	2000	25	3332.00	0.95
100	2000	75	1929.00	0.51
Grand Average			14635.39	15.72

Table 14: Empirical Results, Test Set B

Prob. No.	Nodes	Arcs	Final Objective Function Value IPA-NET	Time (Seconds) IPA-NET	% Maximum Optimality *IPA-gap*
1	1000	2000	6247575167	0.24	+0.00
2	2000	4000	12352067311	0.51	+0.00
3	3000	6000	18230556071	3.76	+0.00
4	4000	8000	23818376116	3.62	+0.00
5	5000	10000	31158650317	4.23	+0.00

contributes an efficient new modeling and solution technology to practitioner's network optimization "tool-box."

Clearly, the IPA principles articulated above can be extended to other categories of interval-flow network problems. Our future research develops this solution approach for more general IFNs.

References

[1] Richard S. Barr and J. Scott Turner. Microdata file merging through large-scale network technology. *Mathematical Programming Study*, 15:1–22, 1981.

[2] R.S. Barr, F. Glover, and D. Klingman. Enhancements of spanning tree labeling procedures for network optimization. *INFOR*, 17:16–34, 1979.

[3] C.F. Bazlamacci and K.S. Hindi. Enhanced adjacent extreme point search and tabu search for the minimum, concave-cost uncapacitated transshipment problem. *Journal of the Operational Research Society*, 47(9):1150–1165, 1996.

[4] Jennifer A. Blue and Kristin P. Bennett. Hybrid extreme point tabu search. *European Journal of Operational Research*, 106(2–3):676–688, 1998.

[5] G.B. Dantzig. *Linear Programming and Extensions.* Princeton University Press, Princeton, NJ, 1963.

[6] L.R. Ford, Jr. and D.R. Fulkerson. *Flows in Networks.* Princeton University Press, Princeton, NJ, 1962.

[7] Fred Glover. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8:156–166, 1977.

[8] Fred Glover and Manuel Laguna. *Tabu Search.* Kluwer Academic Publishers, Boston, 1997.

[9] F.L. Hitchcock. The distribution of a product from several sources to numerous localities. *Journal of Mathematics and Physics*, 20:224–232, 1941.

[10] L.V. Kantorovich. Mathematical methods in the organization and planning of production, translated. *Management Science*, 6:366–422, 1960. (Translation of 1939 article, from the Russian.).

[11] J.P. Kelly, B.L. Golden, and A.A. Assad. Large-scale controlled rounding using tabu search and strategic oscillation. *Annals of Operations Research*, 41:69–84, 1993.

[12] D. Klingman, A. Napier, and J. Stutz. NETGEN: A program for generating large scale capacitated assignment, transportation, and minimum cost flow network problems. *Management Science*, 20:814–821, 1974.

[13] Arne Løkketangen and Fred Glover. Solving zero-one mixed integer programming problems using tabu search. *European Journal of Operational Research*, 106(2–3):624–658, 1998.

[14] J.M. Mulvey. Pivot strategies for primal-simplex network codes. *Journal of the Association for the Computing Machinery*, 25:266–270, 1978.

[15] R.G. Parker and R.L. Rardin. *Discrete Optimization*. Academic Press, New York, NY, 1988.

[16] Minghe Sun, Jay E. Aronson, Patrick G. McKeown, and Dennis Drinka. Hybrid extreme point tabu search. *European Journal of Operational Research*, 106(2–3):441–456, 1998.

[17] W.E. Walker. A heuristic adjacent extreme point algorithm for the fixed charge problem. *Management Science*, 2:587–596, 1976.

[18] J. Whitler. Private communication, 1994.