Algorithmic Operations Research

2-Commodity Integer Network Synthesis Problem

S. N. Kabadi, R. Chandrasekaran et K. P.K. Nair

Volume 4, numéro 2, fall 2009

URI : https://id.erudit.org/iderudit/aor4_2art04

Aller au sommaire du numéro

Éditeur(s)

Preeminent Academic Facets Inc.

ISSN

1718-3235 (numérique)

Découvrir la revue

Citer cet article

Kabadi, S. N., Chandrasekaran, R. & Nair, K. P. (2009). 2-Commodity Integer Network Synthesis Problem. *Algorithmic Operations Research*, *4*(2), 117–132.

Résumé de l'article

We consider the following 2-commodity, integer network synthesis problem: Given two n×n*, non-negative, symmetric, integer-valued matrices* R = (rij) *and* S = (sij) *of minimum flow requirements of* 2 *different commodities, construct an undirected network* G = [N, E, c] *on node set* N = {1, 2, . . . , n} *with integer edge capacities* {c(e) : e ∈ E}*, such that: (i) for any two pairs* (i, j) *and* (k, l), i ≠ j, k ≠ l, *of nodes in* N*, we can simultaneously send* rij *units of flow of commodity* 1 *from* i *to* j *and* skl *units of flow of commodity* 2 *from* k *to* l *in* G*; and (ii)* z = Σ {c(e) : e ∈ E} *is minimum. We present strongly polynomial, combinatorial algorithms for certain special cases of the problem; and for the general problem, we present a strongly polynomial, combinatorial algorithm that produces a feasible solution with objective function value no more than (the optimal objective function value +3).*

Algorithmic Operations Research Vol.4 (2009) 117–132

2-Commodity Integer Network Synthesis Problem

S.N. Kabadi

Faculty of Business Administration, University of New Brunswick, Fredericton, N.B., Canada E3B5A3

R. Chandrasekaran

Department of Computer Science, University of Texas at Dallas, Richardson, Texas, U.S.A.

K.P.K Nair

Faculty of Business Administration, University of New Brunswick, Fredericton, N.B., Canada E3B5A3

Abstract

We consider the following 2-commodity, integer network synthesis problem: Given two $n \times n$, non-negative, symmetric, integer-valued matrices $R = (r_{ij})$ and $S = (s_{ij})$ of minimum flow requirements of 2 different commodities, construct an undirected network $G = [N, E, c]$ on node set $N = \{1, 2, \ldots, n\}$ with integer edge capacities $\{c(e) : e \in E\}$, such that: (i) for any two pairs (i, j) and (k, l), $i \neq j$, $k \neq l$, of nodes in N, we can simultaneously send r_{ij} units of flow of commodity 1 from i to j and s_{kl} units of flow of commodity 2 from k to l in G; and (ii) $z = \sum\{c(e) : e \in E\}$ is minimum. We present strongly polynomial, combinatorial algorithms for certain special cases of the problem; and for the general problem, we present a strongly polynomial, combinatorial algorithm that produces a feasible solution with objective function value no more than (the optimal objective function value $+3$).

Key words: 2-Commodity Flow, Network Synthesis, Strongly Polynomial Algorithm

1. Introduction

Gomory and Hu [7] and Mayeda [17] have considered the following continuous, single-commodity network synthesis problem.

Given an integer $n > 1$ and a symmetric, non-negative, $n \times n$ matrix R, (with $r_{ii} = 0\ \forall i = 1, \ldots, n$), of minimum flow requirements between all pairs of distinct nodes in the node set $N = \{1, 2, \ldots, n\}$, construct an undirected network $G = [N, E, c]$ on node set N with edge set E and non-negative, real-valued edge capacities $\{c(e) : e \in E\}$, such that (i) all the minimum flow requirements are met one at a time, (that is, for any $i, j \in N$, $i \neq j$, the maximum flow value in G from i to j is at least r_{ij}), and (ii) $\sum\{c(e) : e \in E\}$ is minimum.

* The work is supported in part by research grants from the Natural Sciences and Engineering Research Council of Canada to S.N. Kabadi and K.P.K. Nair.
Email: S.N. Kabadi [kabadi@unb.ca], R. Chandrasekaran [x@y.com], K.P.K Nair [nairk@unb.ca].

In both [7] and [17], efficient combinatorial algorithms are presented for the problem. The Gomory-Hu algorithm in [7] has a computational complexity of $O(n^2)$, and when all the elements of the matrix R are integers, the edge capacities in the final network produced by the algorithm are multiples of half. Alternate combinatorial algorithms for the problem are presented in [6,21].

In [3] and [20], an integer version of the single-commodity network synthesis problem is considered. Here, all the elements of the matrix R are integers and the edge capacities of the resultant network are required to be integers. In [3] and independently in [20], algorithms of computational complexity $O(n^2)$ are presented for the problem, and it is shown that whenever $\max\{r_{i,j} : j \in N - \{i\}\} > 1\ \forall i \in N$, the problem has **integer rounding property**, (that is, the difference between the sum of edge capacities in the optimal networks for the integer and continuous versions of the problem is less than 1). (As pointed out in [19], the algorithm in [3] is lacunary and does not apply to some cases.) Alternate algorithms for the problem are given

in [14,19].

These results have been extended to different types of flows such as (i) multipath flows [1,2,11,15,16], (ii) flows with constraints on lengths of paths carrying flow, (these are popularly known as *hop constraints*) [5,10,12,13,18], and (iv) 2-commodity flows [8].

In this paper, we consider the following generalization of the integer, single-commodity network synthesis problem to 2-commodity, integer network synthesis problem (2-INSP):

Given an integer $n > 1$ and two $n \times n$, non-negative, symmetric, integer-valued matrices $R = (r_{ij})$ and $S = (s_{ij})$, (with $r_{ii} = s_{ii} = 0\ \forall i = 1, 2, \ldots, n$), of minimum flow requirements of 2 different commodities, construct an undirected network $G = [N, E, c]$ with integer edge capacities $\{c(e) : e \in E\}$, such that: (i) For any two pairs (i, j) and $(k, l),\ i \neq j,\ k \neq l$, of nodes in N, we can simultaneously send r_{ij} units of flow of commodity 1 from i to j and s_{kl} units of flow of commodity 2 from k to l in G, (it may be noted that we allow $\{i, j\}$ to be equal to $\{k, l\}$); and (ii) $z = \sum\{c(e) : e \in E\}$ is minimum.

When $S = 0$, this problem reduces to the integer, single-commodity case considered in [3,20]. It may be noted that in case of 2-commodity flows, integrality of edge capacities does not guarantee existence of a 2-commodity flow of given integral total flow value with integer edge-flows. It guarantees only half integral edge-flows [9]. In our problem we require only the the edge capacities (and not the edge-flows) to be integers.

As we show in Section 3, an optimal solution to the continuous version of this problem, (where we allow edge capacities to be non-negative reals), can be obtained by solving separately continuous, single-commodity network synthesis problems on matrices R and S and superposing the resultant networks. However, the same approach does not work for 2-INSP. We present efficient optimal schemes for various special cases of the 2-INSP problem. For the general case of this problem, we present a scheme that is guaranteed to produce a solution with sum of edge capacities no more than (the optimal objective function value $+3$).

The following related problem is considered in [8]:

Given an integer $n > 1$ and a symmetric, integral, non-negative, $n \times n$ matrix R of minimum flow requirements, construct an undirected network $G = [N, E, c]$ with integer edge capacities $\{c(e) : e \in E\}$ such that: (i) for any two pairs of distinct nodes (i, j) and (k, l), such that $i \neq j$, $k \neq l$, and $\{i, j\} \neq \{k, l\}$, we can simultaneously pass r_{ij} units of flow from i to j and r_{kl} units of flow from k to l in the network; and (ii) $\sum\{c(e) : e \in E\}$ is minimum. In [8], some interesting structural results are obtained on this problem and an algorithm is presented for the special case of the problem with $r_{ij} = 0/1$.

It may be noted that if corresponding to a given instance of this problem we construct an obvious instance of the 2-INSP problem by defining $S = R$, then in 2-INSP, we allow $\{i, j\} = \{k, l\}$, whereas in the other problem we do not. Thus the two problems are different. We do not know the exact nature of relationship between the two problems. But neither seems to be a special case of the other.

In Section 2, we present minor modifications of (i) one of the algorithms in [6] for the continuous, single-commodity network synthesis problem and (ii) the algorithm in [20] for the integer, single-commodity network synthesis problem. These are used in our algorithms in sections 4 and 5 for 2-INSP. In Section 3, we establish a lower bound for the optimal objective function value of 2-INSP. In Section 4, we present strongly polynomial, combinatorial algorithms for certain special cases of 2-INSP. Finally, in Section 5, we present our algorithm for the general case of the problem that produces a solution with objective function value no more than (OPT + 3), where OPT is the optimal objective function value of the problem.

2. Algorithms For Single-Commodity Network Synthesis Problem

We present in this section (i) a minor modification of an algorithm in [6] for the continuous version of the single-commodity network synthesis problem, which we call the ModG-algorithm, and (ii) a minor modification of the algorithm in [20] for the integer, single-commodity network synthesis problem, which we call the SC-algorithm.

First, we prove a minor result.

Lemma 1 *The respective optimal objective function values of the continuous and integer versions of the single-commodity network synthesis problem remain the same even if we allow the final network to have an additional (Steiner) $(n+1)^{th}$ node. If the additional $((n+1)^{th})$ node is required to be a non-isolated node, then the respective optimal objective function values of the problems with the Steiner node are strictly greater than those of the corresponding problems without the Steiner node.*

Proof. Allowing the $(n+1)^{th}$ node, (which could possibly be an isolated node), will obviously not increase the optimal objective function value. Consider a feasible solution $\overline{G} = [\overline{N}, \overline{E}, \overline{c}]$ to the continuous or integer version of the problem with $\overline{N} = N \cup \{n+1\}$.

If the node $(n+1)$ is isolated in $\overline{G}$, then deleting the $(n+1)^{th}$ node from $\overline{G}$ gives us a feasible solution to the problem without the Steiner node with the same objective function value.

Suppose $\overline{E}$ contains an edge $(n+1, i)$ with positive capacity. Contract this edge in $\overline{G}$ and label the new node i. Replace each pair of parallel edges by a single edge with capacity equal to the sum of capacities of the two edges. Let the resultant network (with node set N) be G'. For any $\{i\} \subseteq X \subset N$, capacity of the cut $(X, N-X)$ in G' is the same as the capacity of the cut $(X \cup \{n+1\}, N-X)$ in $\overline{G}$. Feasibility of the network G' for the problem now follows from the feasibility of the network $\overline{G}$ and the classical max-flow min-cut theorem [4]. The sum of capacities of edges in G' is *strictly* less than the sum of capacities of edges in $\overline{G}$. The result is thus proved. ∎

Using Lemma 1, we assume throughout the rest of this section that in continuous as well as integer single commodity network synthesis problems, each row of the input matrix R contains at least one positive element. (We delete rows/columns of R with all zero elements.)

ModG-algorithm

For each $i \in N = \{1, 2, \ldots, n\}$, let us define $a_i = \max\{r_{ij} : j \in N - \{i\}\}$. Input to the algorithm is the non-negative vector $(a_i : i \in N)$; and the algorithm constructs an optimal solution (network) G^* to the problem such that for any $i, j \in N,\ i \neq j$, we can pass in G^* $\min\{a_i, a_j\} \geq r_{ij}$ units of flow from i to j. Using Lemma 1, we assume, without loss of generality, that $a_i > 0\ \forall i$.

The algorithm orders the nodes in N such that $a_1 = a_2 \geq a_3 \geq \cdots \geq a_n$. In each iteration k, it chooses the subset of nodes $\{1, 2, \ldots, n_k\}$ with the highest current a_i value, peels off the largest constant value Δ^k from the current a_i values of these nodes such that the ordering of the values is preserved, and assigns an additional $\frac{1}{2}\Delta^k$ capacity to each edge in the set $\{(1,2), (2,3), \ldots, (n_k - 1, n_k), (n_k, 1)\}$, (except when $n_k = 2$, in which case the algorithm assigns additional Δ^k capacity to the edge $(1,2)$). These additional edge capacities allow us to send an additional Δ^k units of flow between each pair of nodes in the set $\{1, 2, \ldots, n_k\}$. The process is repeated until all the a_i values are reduced to zero. A formal description of the algorithm is given below.

ModG-algorithm

Step 0: Reorder the nodes in N if necessary such that $a_1 = a_2 \geq a_3 \geq \cdots \geq a_n$.
Set $a_i^0 = a_i\ \forall i \in N$.
Initialize $c^*((i,j)) = 0\ \forall i, j \in N,\ i \neq j$; $k = 0$.

Step 1: If $a_1^k = 0$, go to Step 3.
Else, let n_k be the largest integer such that $a_{n_k}^k = a_1^k$.
If $n_k = n$, then set $\Delta^k = a_1^k$; else,
set $\Delta^k = (a_{n_k}^k - a_{n_k+1}^k)$.
If $n_k = 2$, increase by $c^*((1,2))$ by Δ^k.
If $n_k > 2$, increase $c^*(e)$ by $\frac{1}{2}\Delta^k\ \forall e \in \{(1,2), (2,3), \ldots, (n_k - 1, n_k), (n_k, 1)\}$.
Define $a_i^{k+1} = \begin{cases} a_i^k - \Delta^k & \text{for } i = 1, 2, \ldots, n_k \\ a_i^k & \text{otherwise} \end{cases}$

Step 2 :Increase k by 1, and go to Step 1.

Step 3 :Let $E^* = \{e : c^*(e) > 0\}$.
Output $G^* = [N, E^*, c^*]$ and stop.

Lemma 2 *The output $G^* = [N, E^*, c^*]$ of the ModG-algorithm is connected, contains $O(n)$ number of edges and is a feasible solution to the continuous version of the single-commodity network synthesis problem.*

Though Lemma 2 follows easily from results in [6], we give here a complete proof of the lemma since it will be useful in understanding proofs of validity of algorithms in Section 4. which use the ModG-algorithm as a subroutine.

Proof. Since $a_i > 0\ \forall i$, the set E^* contains edges $\{(1,2), (2,3), \ldots, (n-1, n), (n, 1)\}$. Hence, G^* is connected.

Each edge in E^* belongs to one of the cycles $\{(1, 2, \ldots, j, 1) : j = 3, 4, \ldots, n\}$. The total number of such edges is less than $2n$. The total number of edges in E^* is thus $O(n)$.

By the classical max-flow min-cut theorem [4], it follows that to prove feasibility of the network G^*, it is sufficient to show that for any cut $(X, \overline{X})$ in G^*,

$$c^*[X, \overline{X}] = \sum\{c^*((i,j)) : i \in X; j \in \overline{X}\}$$
$$\geq \min\{\max\{a_i : i \in X\}, \max\{a_i : i \in \overline{X}\}\}.$$

Without loss of generality, let us assume that $1 \in X$. Let $1 < j_1 < j_2 < \cdots < j_l = n+1$ be such that $X = \{1, 2, \ldots, j_1 - 1\} \cup \{j_2, j_2 + 1, \ldots, j_3 - 1\} \cup \cdots$ and $\overline{X} = \{j_1, j_1 + 1, \ldots, j_2 - 1\} \cup \{j_3, j_3 + 1, \ldots, j_4 - 1\} \cup \cdots$.

Obviously, $l \geq 2$. It is easy to see that $\min\{\max\{a_i : i \in X\}, \max\{a_i : i \in \overline{X}\}\} = a_{j_1}$. Thus, we have to show that $c^*[X, \overline{X}] \geq a_{j_1}$.

Let k' be the smallest integer (iteration number) such that $n_{k'} \geq j_1$, and let $\overline{k}$ be the last iteration number of the algorithm. (Thus, $n_{\overline{k}} = n$.) The cut $(X, \overline{X})$ contains $2\lfloor \frac{l}{2} \rfloor$ edges in the cycle $(1, 2, \ldots, n, 1)$, and at least 2 edges in each of the cycles $\{(1, 2, \ldots, n_k, 1) : k = k', k'+1, \ldots, \overline{k} - 1\}$. Hence, in the $\overline{k}^{th}$ iteration, capacities of $2\lfloor \frac{l}{2} \rfloor$ edges in the cut are increased by $\frac{1}{2}\Delta^{\overline{k}}$ each; and for each $k \in \{k', k'+1, \ldots, \overline{k} - 1\}$, if $n_k = 2$, then edge $(1, 2)$ is in the cut and in the k^{th} iteration, its capacity is increased by Δ^k; while if $n_k > 2$, then in the k^{th} iteration, capacities of at least two edges in the cut are increased by $\frac{1}{2}\Delta^k$ each. Hence,

$$c^*[X, \overline{X}] \geq \sum\{\Delta^k : k = k', k'+1, \ldots, \overline{k} - 1\} + \lfloor \tfrac{l}{2} \rfloor \Delta^{\overline{k}} = a_{j_1} + \lfloor \tfrac{l-2}{2} \rfloor \Delta^{\overline{k}} \geq a_{j_1}.$$

This proves the result. ■

Theorem 3 *The ModG-algorithm is a strongly polynomial, combinatorial algorithm with computational complexity $O(n^2)$. The output $G^* = [N, E^*, c^*]$ of the algorithm is an optimal solution to the continuous version of the single-commodity network synthesis problem with*

$$\sum\{c^*(e) : e \in E^*\} = \frac{1}{2}\sum\{a_i : i \in N\}$$

When the input $\{a_i : i \in N\}$ to the algorithm is integer valued, all the edge capacities $\{c^*(e) : e \in E^*\}$ are multiples of half, and when all the a_i values are even, the edge capacities are integers.

Theorem 3 follows easily from results in [6].

SC-algorithm for the integer, single-commodity network synthesis problem

Input to this algorithm is an integer, non-negative, symmetric, $n \times n$ matrix R of minimum flow requirements. It follows from Lemma 1 that we can assume, without loss of generality, that every row of R has a non-zero entry. (Else, we can delete the rows/columns of R with all 0 entries.)

The SC-algorithm computes the a_i value corresponding to each row i of R as in ModG-algorithm. It then (i) deletes all the nodes $i \in N = \{1, 2, \ldots, n\}$ with $a_i = 1$ to get a subset of nodes $\overline{N}$ (in Step 0); (ii) designs an optimal network on the node set $\overline{N}$ using as input the vector $(a_i : i \in \overline{N})$ (in steps 1-5); and (iii) adds to this network the deleted nodes (with $a_i = 1$) and approapriate edges with capacity 1 (in Step 6) to get an optimal solution to the entire problem.

To construct the optimal network on node set $\overline{N}$, the SC-algorithm uses the ModG-algorithm for the continuous version of the problem as a subroutine. If all the values $\{a_i : i \in \overline{N}\}$ are even, then the ModG-algorithm produces a solution with integer capacities. If some of the a_i values are odd, then the SC-algorithm (i) pre-processes the a_i values to make sure that a_1 is odd and $\sum\{a_i : i \in \overline{N}\}$ is even (in steps 1-2); (ii) identifies the subset Q of nodes in $\overline{N}$ with odd modified a_i values, and peels off 1 unit from these odd values (in Step 3); (iii) uses the ModG-algorithm to constructs an optimal network with the resultant a_i values for all $i \in \overline{N}$ (which are now all even) as input (in Step 4); (iv) makes up for the decrease in a_i values for all $i \in Q$ by adding to this network a suitable set of $\frac{|Q|}{2}$ edges which cover all the nodes in Q, and assigning to each of these edges a capacity of 1 and finally makes an adjustment to account for the pre-processing (in Step 5).

SC-algorithm

Step 0: Compute $a_i = \max\{r_{ij} : j \in N - \{i\}\}\ \forall i \in N$.
Let $\overline{N} = \{i : a_i > 1\}$.
Order the nodes $\overline{N} = \{1, 2, \ldots, \overline{n}\}$ such that $a_1 = a_2 \geq a_3 \geq \cdots \geq a_{\overline{n}}$.

Step 1: Set $\overline{a}_i = a_i\ \forall i \in \{2, 3, \ldots, \overline{n}\}$.
If $\sum\{a_i : i \in \overline{N}\}$ = odd, then set $\overline{a}_1 = a_1 + 1$.
Else, set $\overline{a}_1 = a_1$.

Step 2: If $\overline{a}_1$ is even and $\overline{a}_i$ is odd for some $i \in \overline{N}$, then set $\overline{a}_1 = \overline{a}_1 + 1, \overline{a}_2 = \overline{a}_2 + 1$, and Index = 1.
Else, set Index = 0.

Step 3: Let $Q = \{i : i \in \overline{N};\ \overline{a}_i = odd\}$. (By Step 2, if $Q \neq \varnothing$, then $1 \in Q$.) Let $\overline{a}_i = \overline{a}_i - 1\ \forall i \in Q$. Then, $\overline{a}_1 = \overline{a}_2 \geq \cdots \geq \overline{a}_{\overline{n}}$, and they are all even numbers.

Step 4: Use the ModG-algorithm, with input $\{\overline{a}_i : i \in \overline{N}\}$ to construct a network $\overline{G} = [\overline{N}, \overline{E}, \overline{c}]$. (Since all the $\overline{a}_i$'s are even, $\overline{c}(e) =$ integer $\forall e \in \overline{E}$.)

Step 5: Let $Q = \{[1], [2], \ldots, [|Q|]\}$, where

$[1] < [2] < \cdots < [|Q|]$. For each edge, $e \in T = \{([i], [\frac{|Q|}{2} + i]) : i = 1, 2, \ldots, \frac{|Q|}{2}\}$, if $e \in \overline{E}$, then increase $\overline{c}(e)$ by 1; else, add the edge e to $\overline{E}$ and set $\overline{c}(e) = 1$. If Index = 1, then subtract 1 from $\overline{c}((1, 2))$. If $\overline{N} = N$, then define network $G^* = [N, E^*, c^*]$ as $E^* = \overline{E}$ and $c^* = \overline{c}$ and go to Step 7.

Step 6: Construct the graph $G_R = [N, E_R]$, where $E_R = \{(i, j) : r_{ij} > 0\}$. In G_R, shrink the node set $\overline{N}$ to a single pseudo-node and find a spanning forest in the resultant network. In $\overline{G}$, augment the node set to N; and add to $\overline{E}$ edges in G_R corresponding to the edge set of the chosen spanning forest and assign a unit capacity to each of these new edges . Let the resultant network be $G^* = [N, E^*, c^*]$.

Step 7: Output the network $G^* = [N, E^*, c^*]$ and stop.

Theorem 4 *The SC-algorithm is a strongly polynomial, combinatorial algorithm with computational complexity of $O(n^2)$. Let the number of edges in a spanning forest of the graph obtained from the graph G_R, (defined in Step 6 of the SC-algorithm), by shrinking the node set $\overline{N}$, (defined in Step 0 of the SC-algorithm), to a single pseudo-node be p. Then the network $G^* = [N, E^*, c^*]$ produced by the SC-algorithm contains $O(n)$ number of edges and is an optimal solution to the integer, single-commodity network synthesis problem with*

$$\sum\{c^*(e) : e \in E^*\} = p + \left\lceil \frac{1}{2} \sum\{a_i : i \in \overline{N}\} \right\rceil .$$

Also, the subset $\overline{N}$ of nodes, (defined in Step 0 of the algorithm), is connected in G^*.

Theorem 4 follows easily from results in [20]. It may be noted that when $a_i > 1\ \forall i \in N$, the SC-algorithm does not perform Step 6. In such a case, the algorithm actually requires as input only the vector $(a_i : i \in N)$. Hence, *in our algorithms in sections 4 and 5, whenever we use the SC-algorithm as a subroutine with all a_i's greater than 1, we use as input to the algorithm only the vector $(a_i : i \in N)$.*

3. A Lower Bound For The Optimal Objective Function Value of 2-INSP Problem

We now consider the 2-commodity, integer network synthesis problem (2-INSP), defined in Section 1.. The input to the problem consists of two $n \times n$, integer, symmetric, non-negative matrices R and S. Let us denote the optimal objective function value of the problem by OPT.

The following lemma can be proved along the same lines as Lemma 1 using the two-commodity max-flow min-cut theorem of Hu [9].

Lemma 5 *The respective optimal objective function values of 2-INSP and its continuous version remain the same even if we allow the final network to have an additional (Steiner) $(n+1)^{th}$ node. If the additional node is required to be a non-isolated node, then the respective optimal objective function values of the problems with the Steiner node are strictly greater than those of the corresponding problems without the Steiner node.*

For each $i \in N = \{1, 2, \ldots, n\}$, let $a_i = \max\{r_{ij} : j \in N - \{i\}\}$ and $b_i = \max\{s_{ij} : j \in N - \{i\}\}$. *Using Lemma 5 we shall henceforth assume, without loss of generality, that $a_i + b_i > 0\ \forall i \in N$. (Else, delete nodes with $a_i + b_i = 0$.)*

Let us partition the node set N into the following subsets:

$$\begin{aligned} N^{0,1} &= \{i : a_i = 0, b_i = 1\}; \\ N^{1,0} &= \{i : a_i = 1, b_i = 0\}; \\ N^{0,2} &= \{i : a_i = 0, b_i > 1\}; \\ \\ N^{1,1} &= \{i : a_i = b_i = 1\}; \\ N^{1,2} &= \{i : a_i = 1, b_i > 1\}; \\ N^{2,0} &= \{i : a_i > 1, b_i = 0\}; \\ \\ N^{2,1} &= \{i : a_i > 1, b_i = 1\}; \\ N^{2,2} &= \{i : a_i > 1, b_i > 1\} \end{aligned}$$

Let $\tilde{G} = [N, \tilde{E}]$, where $\tilde{E} = \{(i, j) : r_{ij} + s_{ij} > 0\}$, and let $\overline{N} = \{i : a_i + b_i > 1\} = N - \{N^{1,0} \cup N^{0,1}\}$. Shrink in $\tilde{G}$ the node set $\overline{N}$ to a pseudo-node s to get a graph $\overline{G}$. Let the number of edges in a spanning forest of the graph $\overline{G}$ be p.

Lemma 6 *Let OPT' be the optimal objective function value of the instance of the 2-INSP problem on $\overline{N}$ with the corresponding submatrices of R and S as input. Then a lower bound on OPT, the optimal objective function value of 2-INSP on N, is (p + OPT'). If there exists an optimal solution to the problem on $\overline{N}$ in which the entire node set $\overline{N}$ is connected, then an optimal solution to problem on N can be obtained by adding to*

this optimal solution to the problem on $\overline{N}$, edges in $\tilde{G}$ corresponding to edge set of any spanning forest of $\overline{G}$. Thus, in this case, the lower bound is achieved.

Proof. If $\overline{N} = \varnothing$, then the result can be easily seen to be true. (In this case, the elements of R and S merely specify the node-connectivity requirements. The result follows from this obviously.) If the result is not true in general, then let n^* be the minimum value of $|\overline{N}|$ for which a counter-example exists and for this value of $|\overline{N}|$, let p^* be the minimum value of p for which a counter-example exists. Consider a counter-example with input matrices R and S for which $|\overline{N}| = n^*$, and $p = p^*$. Let $G^* = [N, E^*, c^*]$ be an optimal solution to this instance of 2-INSP.

If $p^* = 0$, then using Lemma 5, we can assume that $N = \overline{N}$ and therefore, such an instance of the problem cannot be a counter-example. Hence p^* must be greater than zero.

Let $u \in N - \overline{N}$ be a tip node, (a node of degree 1), of some spanning forest of the graph $\overline{G}$; and let (u, v) be the edge of $\tilde{G}$ corresponding to the edge incident to node u in this spanning forest. Let us assume that $u \in N^{1,0}$. (The other case follows similarly.) Define $(n-1)\times(n-1)$ matrices R^u and S^u, with rows/columns indexed by the set $N^u = N - \{u\}$ as :

$$s^u_{ij} = s_{ij} \ \ \forall \ \ i, j \in N^u;$$

and

$$r^u_{ij} = \begin{cases} r_{ij} & if\ i, j \in N - \{u, v\} \\ \max\{r_{uj}, r_{vj}\} & if\ i = v \text{ and } j \in N - \{u, v\} \\ \max\{r_{iu}, r_{iv}\} & if\ j = v \text{ and } i \in N - \{u, v\} \\ 0 & if\ i = j = v \end{cases}$$

Let $a^u_i = \max\{r^u_{ij} : j \in N^u - \{i\}\}$ and $b^u_i = \max\{s^u_{ij} : j \in N^u - \{i\}\}\ \forall i \in N^u$. Then, $a^u_i = a_i$ and $b^u_i = b_i\ \forall i \in N^u$. Let $OPT(u)$ be the optimal objective function value of the instance of the 2-INSP problem on N^u with input matrices R^u and S^u; and let the corresponding value of p be p^u.

By standard results in network flows [4], it follows that for any node $j \notin \{u, v\}$ of G^*, the maximum flow value from v to j in G^* is at least $\min\{r_{uv}, r_{uj}\}$. Since $u \in N^{1,0}$, this implies that the maximum flow value from v to j in G^* is at least r^u_{vj}. The network G^* is thus a feasible solution to the instance of the 2-INSP problem on N^u with node u as a Steiner node. Also, by the choice of the node u, it follows that G^* contains some edge (u, j) with capacity at least 1. Hence, by Lemma 5, it follows that,

$$OPT = \sum\{c^*(e) : e \in E^*\} \geq OPT(u) + 1$$

Now, $\overline{N^u} = \{i : a^u_i + b^u_i > 1\} = \overline{N}$, and $p^u = p^* - 1$. We thus get,

$$OPT \ \geq \ OPT(u) + 1 \ \geq \ p^* - 1 + \overline{OPT} + 1 \ \geq \ p^* + OPT',$$

where $\overline{OPT}$ is the optimal objective function value of the instance of the 2-INSP problem on $\overline{N^u} = \overline{N}$ with the corresponding submatrices of R^u and S^u as input. Here, the second inequality follows by the definitions of n^* and p^*. The third inequality follows from the fact that each element of the submatrix of R^u (S^u) corresponding to $\overline{N}$ is greater than or equal to the respective element of the submatrix of R (S) corresponding to $\overline{N}$; and hence, any optimal solution to the instance of the 2-INSP problem on $\overline{N}$ with the corresponding submatrices of R^u and S^u as input, is feasible for the the instance of the 2-INSP problem on $\overline{N}$ with the corresponding submatrices of R and S as input.

If there exists an optimal solution to the problem on $\overline{N}$ in which the entire node set $\overline{N}$ is connected, then it is easy to see that the network, obtained by adding to this optimal solution to the problem on $\overline{N}$ edges in $\tilde{G}$ corresponding to edge set of any spanning forest of $\overline{G}$, is feasible to the problem on node set N with objective function value ($p + OPT'$). ■

Lemma 7 $OPT \geq p + \lceil \frac{1}{2} \sum\{a_i + b_i : i \in \overline{N}\} \rceil$.

Proof. It follows from the two-commodity max-flow min-cut theorem of Hu [9] that for any feasible solution $G' = [\overline{N}, E', c']$ to the problem on $\overline{N}$,

$\sum\{c'(i, j) : j \in \overline{N} - \{i\}\} \geq a_i + b_i$ for each $i \in \overline{N}$. Hence,

$2\sum\{c'(e) : e \in E'\} \geq \sum\{(a_i + b_i) : i \in \overline{N}\}$, which implies that,

$\sum\{c'(e) : e \in E'\} \geq \lceil \frac{1}{2} \sum\{(a_i + b_i) : i \in \overline{N}\} \rceil$.

By Lemma 6, we thus get,

$OPT \geq p + \lceil \frac{1}{2} \sum\{(a_i + b_i) : i \in \overline{N}\} \rceil$. ■

4. Algorithms for Special Cases of the 2-INSP Problem

In this section, we develop algorithms that produce optimal solutions for some special cases of the 2-INSP problem. For all the algorithms in this section, we use as input only the non-negative vectors $(a_i : i \in N)$ and $(b_i : i \in N)$, where, for each $i \in N$, $a_i = \max\{r_{ij} : j \in N - \{i\}\}$ and $b_i = \max\{s_{ij} : j \in N - \{i\}\}$, and R

and S are the integer, symmetric, non-negative matrices of minimum flow requirements of the two commodities. *Using Lemma 5, we assume throughout that* $a_i + b_i > 0\ \forall i \in N$.

If $|N| = n = 2$, then obviously $a_1 = a_2$ and $b_1 = b_2$, and $c^*((1,2)) = a_1 + b_1$ is an optimal solution to the problem. Hence, we assume henceforth that $n > 2$.

To start with, we observe that the continuous version of the 2-INSP problem can be solved easily using the results on the continuous, single-commodity network synthesis problem.

Lemma 8 *An optimal solution to the continuous version of the 2-INSP problem, with input matrices R and S, can be obtained by solving separately continuous, single-commodity network synthesis problems on matrices R and S to obtain optimal networks $G^1 = [N, E^1, c^1]$ and $G^2 = [N, E^2, c^2]$, respectively; and superposing the networks G^1 and G^2 to obtain the final network $G^* = [N, E^*, c^*]$ where $E^* = E^1 \cup E^2$ and*

$$c^*(e) = \begin{cases} c^1(e) + c^2(e) & if\ e \in E^1 \cap E^2 \\ c^1(e) & if\ e \in E^1 - E^2 \\ c^2(e) & if\ e \in E^2 - E^1 \end{cases}$$

Proof. The network G^* is obviously feasible for the continuous 2-INSP problem. Using the same arguments as in the proof of lemma 7 but for the problem on the entire node set N, we get $\frac{1}{2}\sum\{(a_i + b_i) : i \in N\}$ as a lower bound on the optimal objective function value of the continuous 2-INSP problem on N. It follows from Theorem 3 that the network G^* achieves this lower bound. ■

If we solve separately integer, single-commodity network synthesis problems on matrices R and S (using the SC-algorithm) and superpose the resultant networks, then it follows from Theorem 4 that the sum of edge capacities of the final network will be

$$p_1 + p_2 + \left\lceil \frac{1}{2} \sum_{i \in N^1} a_i \right\rceil + \left\lceil \frac{1}{2} \sum_{i \in N^2} b_i \right\rceil,$$

where $N^1 = \{i : a_i > 1\};\ N^2 = \{i : b_i > 1\};\ p_1$ is the number of edges in a spanning forest of the graph obtained from $G^1 = [N, E^1]$, where $E^1 = \{(i,j) : r_{ij} > 0\}$, by shrinking the node set N^1 to a pseudo-node; and p_2 is defined similarly with R replaced by S, N^1 by N^2, and E^1 by $E^2 = \{(i,j) : s_{ij} > 0\}$. It can be easily seen that this can be significantly larger than the lower bound in Lemma 7 when $|N^{1,2} \cup N^{2,1} \cup N^{1,1}|$ is large. We now identify some special cases of 2-INSP for which this scheme produces an optimal solution.

Theorem 9 *If $a_i \neq 1$ and $b_i \neq 1\ \forall i \in N$, then obtaining separately optimal networks for integer, single-commodity network synthesis problems on matrices R and S and superposing the two networks produces a feasible solution to 2-INSP with objective function value within one of the optimal. If at least one of $\sum\{a_i : i \in N\}$ and $\sum\{b_i : i \in N\}$ is even, then the solution obtained is an optimal solution to 2-INSP. If each of the two integer, single-commodity network synthesis problems on matrices R and S is solved using the SC-algorithm, then the computational complexity of the entire scheme is $O(n^2)$.*

Proof. Since each network output by the SC-algorithm contains $O(n)$ number of edges, superposing the two networks takes $O(n)$ time. The computational complexity of the scheme is thus the same as that of the SC-algorithm, which is $O(n^2)$.

The network $G^* = [N, E^*, c^*]$ produced by the scheme is obviously feasible for 2-INSP. It follows from Theorem 4 that

$$\begin{aligned} \sum_{e \in E^*} c^*(e) &= \left\lceil \frac{1}{2} \sum_{i \in N} a_i \right\rceil + \left\lceil \frac{1}{2} \sum_{i \in N} b_i \right\rceil \\ &= \left\lceil \frac{1}{2} \sum_{i \in N} (a_i + b_i) \right\rceil + \alpha, \end{aligned}$$

where $\alpha = 0$ or 1; and $\alpha = 0$ if and only if at least one of $\sum\{a_i : i \in N\}$ and $\sum\{b_i : i \in N\}$ is even. The result now follows from Lemma 7. ■

Theorem 10 *Suppose $a_i + b_i > 1\ \forall i \in N$. Let $N^0 = \{i : a_i > 0; b_i > 0\}$. If $|N^0| \leq 1$, then an optimal solution to 2-INSP can be obtained by solving separately integer, single-commodity network synthesis problems on matrices R and S and superposing the resultant networks. Thus, in this case,*

$$OPT = m^1 + m^2 + \left\lceil \frac{1}{2} \sum_{i \in N^1} a_i \right\rceil + \left\lceil \frac{1}{2} \sum_{i \in N^2} b_i \right\rceil,$$

where $N^1 = \{i : a_i > 1\}$, $N^2 = \{i : b_i > 1\}$, and m^1 and m^2 are the number of nodes with $a_i = 1$ and $b_i = 1$, respectively.

Proof. Solving separately integer, single-commodity network synthesis problems on matrices R and S and superposing the resultant networks obviously produces a feasible solution to the 2-INSP problem. It follows from Theorem 4 that when the problem instance satisfies the conditions of the theorem, the objective

function value of this solution is

$$m^1 + m^2 + \left\lceil \frac{1}{2} \sum_{i \in N^1} a_i \right\rceil + \left\lceil \frac{1}{2} \sum_{i \in N^2} b_i \right\rceil .$$

We now prove that under the conditions of the theorem, this solution is optimal to 2-INSP. Thus, consider any optimal solution $G^* = [N, E^*, c^*]$ to such an instance of the 2-INSP problem.

If E^* contains no edge joining some node in $N^{2,0}$ to some node in $N^{0,2}$, then the subnetworks $G^1 = [N^0 \cup N^{2,0}, E^1, c^1]$ and $G^2 = [N^0 \cup N^{0,2}, E^2, c^2]$ of G^* spanned by node sets $N^0 \cup N^{2,0}$ and $N^0 \cup N^{0,2}$, respectively, are feasible solutions to integer, single-commodity network synthesis problems with input matrices R and S, respectively, and the result follows from Theorem 4.

Suppose E^* contains an edge (u, v), (with capacity at least 1), with $u \in N^{2,0}$ and $v \in N^{0,2}$.

Case (i) : $N^{1,1} = \varnothing$ or at least one of $\sum\{a_i : i \in N\}$ and $\sum\{b_i : i \in N\}$ is odd. Contract the edge (u, v) in G^*, label the new node $(n + 1)$, and replace each pair of parallel edges by a single edge with capacity equal to the sum of capacities of the two edges, to get a network $G' = [N', E', c']$, where $N' = (N - \{u, v\}) \cup \{n+1\}$. Define $(n - 1) \times (n - 1)$ matrices R' and S', with rows/columns indexed by the set N', as:

$$r'_{ij} = \begin{cases} r_{ij} & if\ i, j \in N - \{u, v\} \\ r_{uj} & if\ i = n + 1 \text{ and } j \in N - \{u, v\} \\ r_{iu} & if\ j = n + 1 \text{ and } i \in N - \{u, v\} \\ 0 & if\ i = j = n + 1 \end{cases}$$

and

$$s'_{i,j} = \begin{cases} s_{ij} & if\ i, j \in N - \{u, v\} \\ s_{vj} & if\ i = n + 1 \text{ and } j \in N - \{u, v\} \\ s_{iv} & if\ j = n + 1 \text{ and } i \in N - \{u, v\} \\ 0 & if\ i = j = n + 1 \end{cases}$$

Let $a'_i = \max\{r'_{ij} : j \in N' - \{i\}\}$ and $b'_i = \max\{s'_{ij} : j \in N' - \{i\}\}\ \forall i \in N'$. Then $a'_i = a_i$ and $b'_i = b_i\ \forall i \in N - \{u, v\}$; and $a'_{n+1} = a_u$ and $b'_{n+1} = b_v$. We shall show that G' is feasible for the 2-INSP problem with input matrices $R',\ S'$.

By the two-commodity max-flow min-cut theorem of Hu [9], it follows that to prove this it is sufficient to show that for any cut $(X, N' - X)$ in G',

$c'[X, N' - X] = \sum\{c'((i, j)) : i \in X;\ j \in N' - X\} \geq \max\{r'_{ij} : i \in X; j \in N' - X\} + \max\{s'_{ij} : i \in X; j \in N' - X\}$.

Without loss of generality, let us assume that $(n+1) \in X$. Let $X^* = (X \cup \{u, v\}) - \{n + 1\}$. Then,

$c'[X, N' - X] = c^*[X^*, N - X^*] \geq \max\{r_{ij} : i \in X^*; j \in N - X^*\} + \max\{s_{ij} : i \in X^*; j \in N - X^*\} = \max\{r'_{ij} : i \in X; j \in N' - X\} + \max\{s'_{ij} : i \in X; j \in N' - X\}$,

where the inequality follows from the feasibility of G^* for the given instance of 2-INSP and from the two-commodity max-flow min-cut theorem. Thus, G' is feasible for the 2-INSP problem on N'.

We thus get:

$$\begin{aligned} OPT &= \sum_{e \in E^*} c^*(e) \geq \sum_{e \in E'} c'(e) + 1 \\ &\geq \left\lceil \frac{1}{2} \sum \{(a'_i + b'_i) : i \in N'\} \right\rceil + 1 \\ &= \left\lceil \frac{1}{2} \sum \{(a_i + b_i) : i \in N\} \right\rceil + 1 \\ &\geq m^1 + m^2 + \left\lceil \frac{1}{2} \sum_{i \in N^1} a_i \right\rceil + \left\lceil \frac{1}{2} \sum_{i \in N^2} b_i \right\rceil \end{aligned}$$

where m^1 and m^2 are as defined in the statement of the theorem.

Case (ii) : $|N^{1,1}| = 1$ and both $\sum\{a_i : i \in N\}$ and $\sum\{b_i : i \in N\}$ are even. Let $N^{1,1} = \{x\}$. There must be some edge (x, y) in G^* with positive capacity. Let us assume that $y \in N^{2,0}$. (The other case, when $y \in N^{0,2}$ follows similarly.)

Contract the edge (x, y) in G^*, label the new node n^*, and replace each pair of parallel edges by a single edge with capacity equal to the sum of capacities of the two edges, to get a network $\overline{G} = [\overline{N}, \overline{E}, \overline{c}]$, where $\overline{N} = (N - \{x, y\}) \cup \{n^*\}$.

Define $(n - 1) \times (n - 1)$ matrices $\overline{R}$ and $\overline{S}$, with rows/columns indexed by set $\overline{N}$, as

$$\overline{r}_{ij} = \begin{cases} r_{ij} & if\ i, j \in N - \{x, y\} \\ \max\{r_{xj}, r_{yj}\} & if\ i = n^* \text{ and } j \in N - \{x, y\} \\ \max\{r_{ix}, r_{iy}\} & if\ j = n^* \text{ and } i \in N - \{x, y\} \\ 0 & if\ i = j = n^* \end{cases}$$

and

$$\overline{s}_{ij} = \begin{cases} s_{ij} & if\ i, j \in N - \{x, y\} \\ s_{xj} & if\ i = n^* \text{ and } j \in N - \{x, y\} \\ s_{ix} & if\ j = n^* \text{ and } i \in N - \{x, y\} \\ 0 & if\ i = j = n^* \end{cases}$$

Let $\overline{a}_i = \max\{\overline{r}_{ij} : j \in \overline{N} - \{i\}\}$ and $\overline{b}_i = \max\{\overline{s}_{ij} : j \in \overline{N} - \{i\}\}$. Then $\overline{a}_i = a_i$ and $\overline{b}_i = b_i$ $\forall i \in N - \{x, y\}$; and $\overline{a}_{n^*} = a_y$ and $\overline{b}_{n^*} = b_x = 1$. Let $\overline{N}^1 = (N^1 - \{y\}) \cup n^*$. It can be shown, using arguments similar to those in Case (i), that $\overline{G}$ is feasible for the 2-INSP problem with input matrices $\overline{R}$, $\overline{S}$. The 2-INSP problem with input matrices $\overline{R}$, $\overline{S}$ is of the type discussed in Case (i). Using this, and the validity of the theorem for Case (i), we get:

$$\begin{aligned} OPT &= \sum_{e \in E^*} c^*(e) \geq \sum_{e \in \overline{E}} \overline{c}(e) + 1 \\ &\geq 1 + \left\lceil \frac{1}{2} \sum_{i \in \overline{N}^1} \overline{a}_i \right\rceil + \left\lceil \frac{1}{2} \sum_{i \in N^2} \overline{b}_i \right\rceil + 1 \\ &= 2 + \left\lceil \frac{1}{2} \sum_{i \in N^1} a_i \right\rceil + \left\lceil \frac{1}{2} \sum_{i \in N^2} b_i \right\rceil \\ &= m^1 + m^2 + \left\lceil \frac{1}{2} \sum_{i \in N^1} a_i \right\rceil + \left\lceil \frac{1}{2} \sum_{i \in N^2} b_i \right\rceil \end{aligned}$$

This proves the theorem. ■

We shall now give algorithms for other non-trivial, special cases of 2-INSP.

Case 1 *:* $a_i > 1\ \forall i \in N$.

Input to this algorithm is non-negative vectors $(a_i : i \in N)$ and $(b_i : i \in N)$, the elements of which are defined as before.

In Step 0, the algorithm arranges the elements of N in non-increasing order of their a_i values; obtains an alternate ordering $\{k_1, k_2, \ldots, k_{n_0}\}$ of the subset of nodes with positive b_i values such that the b_i values are non-increasing; and then pre-processes the input data to obtain modified values $\{\tilde{a}_i, \tilde{b}_i : i \in N\}$ such that (i) $\sum\{\tilde{a}_i + \tilde{b}_i : i \in N\}$ is even; (ii) the orderings of the two sets of values are preserved; (iii) if there exists an odd $\tilde{a}_i$-value, then $\tilde{a}_1$ is odd; and (iv) if there exists an odd $\tilde{b}_i$-value, then $\tilde{b}_{k_1}$ is odd.

Next, in Step 1, the algorithm (i) identifies the subsets Q^a and Q^b of nodes with odd $\tilde{a}_i$ and $\tilde{b}_i$ values, respectively; (ii) peels off 1 unit from each of these odd $\tilde{a}_i$ and $\tilde{b}_i$ values to obtain modified values, $\{\overline{a}_i, \overline{b}_i : i \in N\}$; and (iii) defines the multiset $Q = Q^a \cup Q^b$. (Thus, Q contains 2 copies of each of the nodes in $Q^a \cap Q^b$.)

In Step 2, the algorithm uses the ModG-algorithm to constructs optimal networks G^a and G^b (with integer capacities) for even-valued input vectors $(\overline{a}_i : i \in N)$ and $(\overline{b}_i : i \in N)$, respectively, and superposes these two networks to obtain network $\overline{G}$ with integer capacities.

In Step 3, it makes up for the difference of 1 between some of the values in $\{\overline{a}_i, \overline{b}_i : i \in N\}$ and their corresponding values in $\{\tilde{a}_i, \tilde{b}_i : i \in N\}$ by adding to $\overline{G}$ a suitable set of $\frac{1}{2}|Q|$ edges which cover all the nodes in Q, and assigning to each of these edges a capacity of 1; and finally it makes an adjustment to account for the pre-processing in Step 0.

Algorithm 3

Step 0: Let $N^0 = \{i : b_i > 0\}$, and $|N^0| = n_0$.
Arrange nodes in the set $N = \{1, 2, \ldots, n\}$ such that $a_1 = a_2 \geq \cdots \geq a_n$.
Also, find an ordering $\{k_1, k_2, \ldots, k_{n_0}\}$ of the elements of N^0 such that $b_{k_1} = b_{k_2} \geq b_{k_3} \geq \cdots \geq b_{k_{n_0}}$.
If $\sum\{a_i + b_i : i \in N\}$ is odd, then set $\tilde{a}_1 = a_1 + 1$.
Else, set $\tilde{a}_1 = a_1$.
Set $\tilde{a}_i = a_i\ \forall i \in \{2, 3, \ldots, n\}$ and $\tilde{b}_i = b_i\ \forall i \in N$.
If $\tilde{a}_1$ is even, and at least one of the $\tilde{a}_i$'s is odd, then set $\tilde{a}_1 = \tilde{a}_1 + 1, \tilde{a}_2 = \tilde{a}_2 + 1$, and Index1 = 1.
Else, set Index1 = 0.
Similarly, if $\tilde{b}_{k_1}$ is even, and at least one of the $\tilde{b}_i$'s is odd, then set $\tilde{b}_{k_1} = \tilde{b}_{k_1} + 1$, $\tilde{b}_{k_2} = \tilde{b}_{k_2} + 1$, and Index2 = 1.
Else, set Index2 = 0.

Step 1: Let $Q^a = \{i : \tilde{a}_i = odd\}$ and $Q^b = \{i : \tilde{b}_i = odd\}$; and multiset $Q = Q^a \cup Q^b$. (Thus, Q contains 2 copies of each node in $Q^a \cap Q^b$.)
Set $\overline{a}_i = \begin{cases} \tilde{a}_i & if\ i \notin Q^a \\ \tilde{a}_i - 1 & if\ i \in Q^a \end{cases}$
$\overline{b}_i = \begin{cases} \tilde{b}_i & if\ i \notin Q^b \\ \tilde{b}_i - 1 & if\ i \in Q^b \end{cases}$

Step 2: With $\{\overline{a}_i : i \in N\}$ as input, perform the ModG-algorithm to obtain a network $G^a = [N, E^a, c^a]$.
Similarly, with $\{\overline{b}_i : i \in N^0\}$ as input, perform the ModG-algorithm to obtain a network $G^b = [N^0, E^b, c^b]$. Superpose the two networks G^a and G^b to obtain network $\overline{G} = [N, \overline{E}, \overline{c}]$, where, $\overline{E} = E^a \cup E^b$, and $\overline{c}(e) = \begin{cases} c^a(e) & if\ e \in E^a - E^b \\ c^b(e) & if\ e \in E^b - E^a \\ c^a(e) + c^b(e) & if\ e \in E^a \cap E^b \end{cases}$

Step 3: Arrange the elements of $Q = \{[1], [2], \ldots, [|Q|]\}$ such that $[1] \leq [2] \leq \cdots \leq [|Q|]$.

(Thus, if $Q^a \neq \varnothing$, then $[1] = 1$.)
Define $T = \{([i],[\frac{|Q|}{2} + i]) : i = 1,2,\ldots,\frac{|Q|}{2}\}$.
For each edge $e \in T \cap \overline{E}$ add 1 to $\overline{c}(e)$.
For each edge $e \in T - \overline{E}$ add the edge e to $\overline{E}$ and assign it a capacity $\overline{c}(e) = 1$.
Let the resultant network be $\tilde{G} = \{N, \tilde{E}, \tilde{c}\}$.
Let $c^*(e) = \tilde{c}(e)\ \forall e \in \tilde{E}$.
If Index1 = 1, reduce, $c^*((1,2))$ by 1.
If Index2 = 1, reduce $c^*((k_1,k_2))$ by 1.
Let $E^* = \{e \in \tilde{E} : c^*(e) > 0\}$.
Output the network $G^* = [N, E^*, c^*]$ and stop.

Theorem 11 *The network G^* constructed by Algorithm 3 is an optimal solution for Case 1 of 2-INSP.*
Proof. By the two-commodity max-flow min-cut theorem of Hu [9], it follows that to prove the feasibility of the network G^* output by the algorithm, it is sufficient to show that for any cut $(X, \overline{X})$ in G^*,

$$c^*[X,\overline{X}] \geq \min\{\max\{a_i : i \in X\}, \max\{a_i : i \in \overline{X}\}\} + \min\{\max\{b_i : i \in X\}, \max\{b_i : i \in \overline{X}\}\}.$$

Without loss of generality, let us assume that $1 \in X$. Let $1 < j_1 < j_2 < \cdots < j_l = n+1$ be such that $X = \{1,2,\ldots,j_1-1\} \cup \{j_2, j_2+1, \ldots, j_3-1\} \cup \cdots$ and $\overline{X} = \{j_1, j_1+1, \ldots, j_2-1\} \cup \{j_3, j_3+1, \ldots, j_4-1\} \cup \cdots$.

Obviously, $l \geq 2$. It is easy to see that $\min\{\max\{a_i : i \in X\}, \max\{a_i : i \in \overline{X}\}\} = a_{j_1}$; and that there exists $h \neq k_1$ such that $\min\{\max\{b_i : i \in X\}, \max\{b_i : i \in \overline{X}\}\} = b_h$, and nodes h and k_1 lie on different sides of the bipartition $(X, \overline{X})$. Thus, we have to show that $c^*[X,\overline{X}] \geq a_{j_1} + b_h$.

We first show that $\tilde{c}[X,\overline{X}] \geq \tilde{a}_{j_1} + \tilde{b}_h$.

From the proof of Lemma 2, noting that $\Delta^{\overline{k}} \geq 1$, we get,

$$\overline{c}[X,\overline{X}] \geq \overline{a}_{j_1} + \overline{b}_h + 2\left\lfloor \frac{l-2}{2} \right\rfloor \tag{1}$$

If $l \geq 4$, then we get using inequality 1:

$$\begin{aligned} \tilde{c}[X,\overline{X}] &\geq \overline{c}[X,\overline{X}] && (2)\\ &\geq \overline{a}_{j_1} + \overline{b}_h + 2 && (3)\\ &\geq \tilde{a}_{j_1} + \tilde{b}_h && (4) \end{aligned}$$

Suppose $2 \leq l \leq 3$. Then nodes 1 and j_1 lie on different sides of the bipartition $(X, \overline{X})$ and similarly, nodes k_1 and h lie on different sides of the bipartition. We consider all the four possible cases:

(i). Both $\tilde{a}_{j_1}$ and $\tilde{b}_h$ are even : In this case,

$$\begin{aligned} \tilde{c}[X,\overline{X}] &\geq \overline{c}[X,\overline{X}] && (5)\\ &\geq \overline{a}_{j_1} + \overline{b}_h && (6)\\ &= \tilde{a}_{j_1} + \tilde{b}_h && (7) \end{aligned}$$

Here, the second inequality follows from expression 1.

(ii). $\tilde{a}_{j_1}$ is odd and $\tilde{b}_h$ is even : In this case, $\tilde{a}_1$ is odd and the edge set T contains edges $(1,u)$ and (j_1,v), for some $u, v \in Q$. If $u \in \overline{X}$, then the edge $(1,u)$ is in the cut $(X,\overline{X})$. Else, if $u < j_1$, then by the definition of the set T, $v < u < j_1$; and if $u \geq j_2$, then, again by the definition of the set T, $v > u \geq j_2$. In either case, the edge (j_1,v) is in the cut $(X,\overline{X})$. From this, and inequality 1, we get:

$$\begin{aligned} \tilde{c}[X,\overline{X}] &\geq \overline{c}[X,\overline{X}] + 1 && (8)\\ &\geq \overline{a}_{j_1} + \overline{b}_h + 1 && (9)\\ &= \tilde{a}_{j_1} + \tilde{b}_h && (10) \end{aligned}$$

(iii). $\tilde{b}_h$ is odd and $\tilde{a}_{j_1}$ is even : In this case, $\tilde{b}_{k_1}$ is odd, and it can be shown using arguments similar to those in case (ii) that expression 10 above holds.

(iv). Both $\tilde{a}_{j_1}$ and $\tilde{b}_h$ are odd : In this case, both $\tilde{a}_1$ and $\tilde{b}_{k_1}$ are odd. Let $\{y,z\} = \{k_1,h\}$, where $y \in X$ and $z \in \overline{X}$. Then the edge set T, (defined in Step 3 of the algorithm), contains edges $(1,u)$, (j_1,v), (y,w) and (z,x), for some u, v, w and x in Q.

If any one of the edges $(1,u)$ and (y,w) is not in the cut $(X,\overline{X})$, then it can be shown using the same arguments as those in case (ii) that each of the edges (j_1,v), and (z,x) is in the cut. Thus, at least two edges of the edge-set T lie in the cut $(X,\overline{X})$. Using inequality 1, we get:

$$\begin{aligned} \tilde{c}[X,\overline{X}] &\geq \overline{c}[X,\overline{X}] + 2 && (11)\\ &\geq \overline{a}_{j_1} + \overline{b}_h + 2 && (12)\\ &= \tilde{a}_{j_1} + \tilde{b}_h && (13) \end{aligned}$$

Now, $c^*((1,2)) < \tilde{c}((1,2))$ iff $a_1 < \tilde{a}_1$; and $c^*((k_1,k_2)) < \tilde{c}((k_1,k_2))$ iff $b_{k_1} < \tilde{b}_{k_1}$. From this and expressions 4, 7, 10 and 13, we get :

$$c^*[X,\overline{X}] \geq a_{j_1} + b_h$$

This proves the feasibility of G^*.

Using Theorem 3, we get :

$$\begin{aligned}\sum_{e\in E^*} c^*(e) &= \sum_{e\in E^a} c^a(e) + \sum_{e\in E^b} c^b(e) + \\ &\quad |T| - Index1 - Index2 \\ &= \frac{1}{2}\sum_{i\in N}\overline{a}_i + \frac{1}{2}\sum_{i\in N}\overline{b}_i + |T| - Index1 - Index2 \\ &= \left\lceil \frac{1}{2}\sum_{i\in N}(a_i + b_i) \right\rceil\end{aligned}$$

The optimality of G^* now follows from Lemma 7. ■

Case 2 *: $a_i > 0$ and $b_i > 0 \ \ \forall i \in N$.*

Input to this algorithm is positive vectors $(a_i : i \in N)$ and $(b_i : i \in N)$. Here, we use the fact that if we add 1 to each a_i value and subtract 1 from each b_i value and the new values are non-negative, then a network is feasible for the old $a_i,\ b_i$ values if and only if it is feasible for the modified $a_i,\ b_i$ values. Thus, we add 1 to each a_i value and subtract 1 from each b_i value. This reduces the problem to Case 1 which we solve using Algorithm 3.

Algorithm 4

Step 1: Set $\overline{a}_i = a_i + 1$ and $\overline{b}_i = b_i - 1\ \forall i \in N$

Step 2: Perform Algorithm 3, with input vectors $(\overline{a}_i : i = 1, 2, \ldots, n)$ and $(\overline{b}_i : i = 1, 2, \ldots, n)$, to construct a network $G^* = [N, E^*, c^*]$. Output the network G^*.

Theorem 12 *The network G^*, constructed by Algorithm 4, is an optimal solution for Case 2 of 2-INSP.*

Proof. Since $a_i > 0$ and $b_i > 0$ for each $i \in N$, both the vectors $(\overline{a}_i : i \in N)$ and $(\overline{b}_i : i \in N)$, defined in Step 1 of Algorithm 4, are non-negative vectors. It follows from Theorem 11 that the network G^* produced by Algorithm 4 is an optimal solution for input vectors $(\overline{a}_i : i \in N),\ (\overline{b}_i : i \in N)$. Also, from the proof of Theorem 11, it follows that for any cut $(X, \overline{X})$ in G^*,

$$\begin{aligned}c^*[X, \overline{X}] &\geq \min\{\max\{\overline{a}_i : i \in X\}, \max\{\overline{a}_i : i \in \overline{X}\}\} \\ &\quad + \min\{\max\{\overline{b}_i : i \in X\}, \max\{\overline{b}_i : i \in \overline{X}\}\} \\ &= \min\{\max\{a_i : i \in X\}, \max\{a_i : i \in \overline{X}\}\} \\ &\quad + \min\{\max\{b_i : i \in X\}, \max\{b_i : i \in \overline{X}\}\}\end{aligned}$$

The feasibility of the network G^* for the given input vectors now follows from the two-commodity max-flow min-cut theorem [9].

Also, using Theorem 11, we get:

$$\sum_{e\in E^*} c^*(e) = \left\lceil \frac{1}{2}\sum_{i\in N}\{\overline{a}_i + \overline{b}_i\} \right\rceil = \left\lceil \frac{1}{2}\sum_{i\in N}\{a_i + b_i\} \right\rceil.$$

The optimality of the solution now follows from Lemma 7. ■

Case 3 *: $a_i + b_i > 1\ \forall i \in N$ and $|\{i : a_i > 0, b_i > 0\}| > 1$.*

Input to this algorithm is symmetric, non-negative, integer matrices R and S of minimal flow requirements of two commodities. We define vectors $(a_i : i \in N)$ and $(b_i : i \in N)$, as before.

For any non-empty, proper subset X of N, let us define $r(X) = \max\{r_{ij} : i \in X; j \in \overline{X}\}$, and $s(X) = \max\{s_{ij} : i \in X; j \in \overline{X}\}$.

The algorithm divides the node set N into three sets : $N^{2,0}$, $N^{0,2}$ and the set of remaining nodes, which is denoted by N^0. As shown at the beginning of this section, if $|N^0| \leq 1$ then an optimal solution to the problem can be obtained by applying the SC-algorithm to the a-vector and the b-vector, separately, and superposing the two networks. Hence, here we only consider the case with $|N^0| > 1$.

If $r_{ij} = 0$ for all $i \in N^0$ and $j \in N^{2,0}$, and at least one of $\sum\{a_i + b_i : i \in N^0\}$ and $\sum\{a_i : i \in N^{2,0}\}$ is even, or if the same situation holds for S with a_i's replaced by b_i's and vice versa, then the algorithm sets index "Case" =1. In this case, optimal network for requirements on node set $N^{2,0}$ is constructed separately using the SC-algorithm; optimal network for requirements on the rest of the nodes is constructed separately and the final solution is the disjoint union of these two networks.

Else, as in Algorithm 4, for each i in N^0 we increase the a_i value by 1 and decrease the b_i value by 1. The a_i and b_i values are further modified to obtain values $\{\overline{a}_i, \overline{b}_i : i \in N\}$, such that (i) each of $\sum\{\overline{a}_i + \overline{b}_i : i \in N^0\} + \sum\{\overline{a}_i : i \in N^{2,0}\}$ and $\sum\{\overline{b}_i : i \in N^{0,2}\}$ is even; (ii) the ordering of each of the four sets of values is preserved; (iii) if there exists an odd $\overline{a}_i$-value ($\overline{b}_i$ value) in $N^0 \cup N^{2,0}$ (N^0), then its largest $\overline{a}_i$ value ($\overline{b}_i$ value) is odd; and (iv) if there exists an odd $\overline{b}_i$ value in $N^{0,2}$, then its largest $\overline{b}_i$ value is odd.

We then (i) identify subsets $Q^{a,0}$ and $Q^{a,1}$ of node sets N^0 and $N^{2,0}$, respectively, with odd $\overline{a}_i$ values;

and subsets $Q^{b,0}$ and $Q^{b,1}$ of nodes in N^0 and $N^{0,2}$, respectively, with odd $\overline{b}_i$ values; (ii) peel off 1 unit from each of these odd $\overline{a}_i$ and $\overline{b}_i$ values, to obtain modified values, $\{\hat{a}_i, \hat{b}_i : i \in N\}$; (iii) further modify the largest $\hat{a}_i$ and $\hat{b}_i$ values in some of the sets N^0, $N^{2,0}$ and $N^{0,2}$ such that the two largest $\hat{a}_i$ values are equal and the two largest $\hat{b}_i$ values are equal.

In steps 3 and 4, the algorithm uses a modification of the ModG-algorithm to construct optimal networks (with integer capacities) for even-valued input vectors $(\hat{a}_i : i \in N)$ and $(\hat{b}_i : i \in N)$, respectively, and superposes these two networks to obtain network $\hat{G}$ with integer capacities.

In Step 5, we make up for the difference of 1 between some of the values in $\{\hat{a}_i, \hat{b}_i : i \in N\}$ and the corresponding values in $\{\overline{a}_i, \overline{b}_i : i \in N\}$ by adding to $\hat{G}$ a suitable set of edges which cover all the nodes in the multiset $Q^{a,0} \cup Q^{a,1} \cup Q^{b,0}$, and the set $Q^{b,1}$, (each of which has even cardinality); and assigning to each of these edges a capacity of 1, to get a network $\overline{G}$.

Finally, in Step 6, we make an adjustment to account for some pre-processing and alterations to a_i and b_i values done previously.

Algorithm 5

Step 0: Set $N^0 = \{i : a_i > 0, b_i > 0\}$;
$Case = Index_0^a = Index^a = 0$;
$Index_I^a = Index_I^b = Index_0^b = 0$;
$Ind_a^f = Ind_b^f = 0$.
Set $n_0 = |N^0|$ and $n_1 = |N^0 \cup N^{2,0}|$.
Number the nodes in N^0 as $\{1, 2, \ldots, n_0\}$ such that $a_1 \geq a_2 \geq \cdots \geq a_{n_0}$.
Find an alternate ordering $\{t_1, t_2, \ldots, t_{n_0}\}$ of nodes in N^0 such that $b_{t_1} \geq b_{t_2} \geq \cdots \geq b_{t_{n_0}}$.
Number the nodes in $N^{2,0}$ as $\{n_0 + 1, n_0 + 2, \ldots, n_1\}$ and number the nodes in $N^{0,2}$ as $\{n_1 + 1, n_1 + 2, \ldots, n\}$, such that $a_i \geq a_j \; \forall \; n_0 < i < j \leq n_1$, and $b_i \geq b_j \; \forall \; n_1 < i < j \leq n$.

Step 1a: If $r(N^0) = \max\{r_{ij} : i \in N^0; j \in \overline{N^0}\} = 0$ and at least one of $\sum\{a_i + b_i : i \in N^0\}$ and $\sum\{a_i : i \in N^{2,0}\}$ is even, then set $Case = 1$ and go to Step $1b$.
If $s(N^0) = \max\{s_{ij} : i \in N^0; j \in \overline{N^0}\} = 0$ and at least one of $\sum\{a_i + b_i : i \in N^0\}$ and $\sum\{b_i : i \in N^{0,2}\}$ is even,then rename the a_i's as b_i's and the b_i's as a_i's; set $Case = 1$; and go to Step $1b$.
If $\sum_{i \in N} a_i + \sum_{i \in N^0} b_i$ is even, then go to Step $1c$.
If $\sum\{b_i : i \in N\} + \sum\{a_i : i \in N^0\}$ is even, then rename all the a_i's as b_i's and all the b_i's as a_i's; and go to Step $1c$.

Step 1b: If $\sum_{i \in N^0}\{a_i + b_i\}$ is odd, then add 1 to b_{t_1}.
If $\sum_{i \in N^{2,0}} a_i$ is odd, then add 1 to a_{n_0+1}.

Step 1c: If $\sum_{i \in N^{0,2}} b_i$ is odd then add 1 to b_{n_1+1}.
Set $\overline{a}_i = a_i + 1$ and $\overline{b}_i = b_i - 1 \; \forall i \in N^0$; and $\overline{a}_i = a_i$, and $\overline{b}_i = b_i$ for every other i.
If $Case = 0$, then go to Step $2b$.

Step 2a: Construct an optimal network $G^{a,1} = [N^{2,0}, E^{a,1}, c^{a,1}]$ using the SC-algorithm with $(\overline{a}_i : i \in N^{2,0})$ as input vector.
If $s(N^0) = 0$ then, construct an optimal network $G^0 = [N^0, E^0, c^0]$ using Algorithm 3 with $(\overline{a}_i : i \in N^0)$ and $(\overline{b}_i : i \in N^0)$ as input vectors. Construct an optimal network $G^{b,1} = [N^{0,2}, E^{b,1}, c^{b,1}]$ using the SC-algorithm with $(\overline{b}_i : i \in N^{0,2})$ as input vector.
Let $G^* = [N, E^*, c^*]$, where, $E^* = E^0 \cup E^{a,1} \cup E^{b,1}$; and

$$c^*(e) = \begin{cases} c^{a,1}(e) & \forall e \in E^{a,1} \\ c^0(e) & \forall e \in E^0 \\ c^{b,1}(e) & \forall e \in E^{b,1} \end{cases}$$

Go to Step 7.
Else,
If $\overline{a}_1$= even, and $\overline{a}_i$ is odd for some $i \in N^0$, then set $Index_0^a = 1$, $\overline{a}_1 = \overline{a}_1 + 1, \overline{a}_2 = \overline{a}_2 + 1$.
Compute $Q^{a,0} = \{i : \overline{a}_i = odd; \; i \in N^0\}\}$.
set $\hat{a}_i = \overline{a}_i - 1 \; \forall i \in Q^{a,0}$; $\hat{a}_i = \overline{a}_i$ for every other $i \in N^0$.
Construct an optimal network $G^{a,0} = [N^0, E^{a,0}, c^{a,0}]$ using the ModG-algorithm with $(\hat{a}_i : i \in N^0)$ as input vector.
Let $\hat{G}^a = [N^0 \cup N^{2,0}, \hat{E}^a, \hat{c}^a]$, where, $\hat{E}^a = E^{a,1} \cup E^{a,0}$; and

$$\hat{c}^a(e) = \begin{cases} c^{a,1}(e) & \forall e \in E^{a,1} \\ c^{a,0}(e) & \forall e \in E^{a,0} \end{cases}$$

Go to Step $4a$.

Step 2b: If $\overline{a}_1 \geq \overline{a}_{(n_0+1)}$, then set $u = 1$.
Else, set $u = n_0 + 1$.
If $\overline{a}_u$= even, and $\overline{a}_i$ is odd for some $i \in N^0 \cup N^{2,0}$, then set $Index^a = 1$, $\overline{a}_u = \overline{a}_u + 1$, $\overline{a}_{(u+1)} = \overline{a}_{(u+1)} + 1$.
Define $Q^{a,0} = \{i : \overline{a}_i = odd; \; i \in N^0\}$;
$Q^{a,1} = \{i : \overline{a}_i = odd; \; i \in N^{2,0}\}$;
Set $\widehat{a}_i = \overline{a}_i - 1 \; \forall i \in Q^{a,0} \cup Q^{a,1}$; $\widehat{a}_i = \overline{a}_i$

for every other i.
Set $\hat{a}_i = \widehat{a}_i \ \forall\ i$.
If $2\lceil \frac{1}{2} r(N^0) \rceil < \min\{\hat{a}_i : i \in N^0 \cup N^{2,0}\}$, then set $Ind_a^f = 1$ and go to Step $3a$.
If $r(N^0) < \min\{a_1, a_{(n_0+1)}\} - 1$, then go to Step $3a$.
If $\hat{a}_{(n_0+1)} = \hat{a}_1$, then set $Index_I^a = 1$.
Else, add 2 to the smaller of $\hat{a}_{(n_0+1)}$ and $\hat{a}_1$.

Step 3a: Set $\hat{c}^a(e) = 0\ \forall e$; $a_i^0 = \hat{a}_i\ \forall i$; and $\ell = f = 0$.

Step 3b: Let $\alpha = \max\{a_i^\ell : i \in N^0 \cup N^{2,0}\}$.
If $\alpha = 0$, go to Step 4a.
If $a_1^\ell < \alpha$, then set $x = 0$; else, let $x \in N^0$ be the largest integer such that $a_x^\ell = \alpha$.
If $a_{(n_0+1)}^\ell < \alpha$, then set $y = n_0$;
else, let $y \in N^{2,0}$ be the largest integer such that $a_y^\ell = \alpha$.
If $x = n_0$ then set $\beta_1 = 0$; else set $\beta_1 = a_{(x+1)}^\ell$.
If $y = n_1$, then set $\beta_2 = 0$;
else set $\beta_2 = a_{(y+1)}^\ell$.
Set $\beta = \max\{\beta_1, \beta_2\}$.
If the set $\{1, 2, \ldots, x\} \cup \{n_0 + 1, n_0 + 2, \ldots, y\}$ contains only 2 elements, say $\{w, z\}$, then increase $\hat{c}^a((w, z))$ by $(\alpha - \beta)$.
Else,
if $y = n_0$, then increase the value of $\hat{c}^a(e)$ by $\frac{1}{2}(\alpha - \beta)$ for every $e \in \{(1, 2), (2, 3) \ldots, (x - 1, x), (x, 1)\}$;
if $x = 0$, then increase the value of $\hat{c}^a(e)$ by $\frac{1}{2}(\alpha - \beta)$ for every $e \in \{(n_0 + 1, n_0 + 2), (n_0 + 2, n_0 + 3), \ldots, (y - 1, y), (y, n_0 + 1)\}$;
otherwise, increase the value of $\hat{c}^a(e)$ by $\frac{1}{2}(\alpha - \beta)$ for every $e \in \{(1, 2), \ldots, (x - 1, x), (x, n_0 + 1), (n_0 + 1, n_0 + 2), \ldots, (y - 1, y), (y, 1)\}$.
Set $a_i^{\ell+1} = \begin{cases} \beta \ \ if \ \ i \in N^0,\ i \le x;\ \text{or} \\ \quad if \ \ i \in N^{2,0},\ i \le y \\ a_i^\ell \ otherwise \end{cases}$
Update $f = f + (\alpha - \beta)$
If $f \ge r(N^0)$ and $Ind_a^f = 1$, then set $\ell = \ell + 1$ and go to Step $3d$.

Step 3c: Set $\ell = \ell + 1$ and go to Step $3b$.

Step 3d: Construct optimal networks $G^{a,0} = [N^0, E^{a,0}, c^{a,0}]$ and $G^{a,1} = [N^{2,0}, E^{a,1}, c^{a,1}]$ using the ModG-algorithm with $(a_i^\ell : i \in N^0)$ and $(a_i^\ell : i \in N^{2,0})$ as input vectors, respectively. Update

$$\hat{c}^a(e) = \begin{cases} \hat{c}^a(e) + c^{a,0}(e)\ \forall e \in E^{a,0} \\ \hat{c}^a(e) + c^{a,1}(e)\ \forall e \in E^{a,1} \\ \hat{c}^a(e) \qquad\qquad otherwise \end{cases}$$

Step 4a: If $\overline{b}_{t_1}$ = even, and the set $\{\overline{b}_i : i \in N^0\}$ contains at least one odd value, then set $Index_0^b = 1, \overline{b}_{t_1} = \overline{b}_{t_1} + 1, \overline{b}_{t_2} = \overline{b}_{t_2} + 1$.
If $\overline{b}_{(n_1+1)}$ = even, and the set $\{\overline{b}_i : i \in N^{0,2}\}$ contains at least one odd value, then set $Index_1^b = 1$, $\overline{b}_{(n_1+1)} = \overline{b}_{(n_1+1)} + 1$, $\overline{b}_{(n_1+2)} = \overline{b}_{(n_1+2)} + 1$.
Define $Q^{b,0} = \{i : \overline{b}_i = odd;\ i \in N^0\}$;
$Q^{b,1} = \{i : \overline{b}_i = odd;\ i \in N^{0,2}\}$.
Set $\widehat{b}_i = \overline{b}_i - 1\ \forall i \in Q^{b,0} \cup Q^{b,1}$; and $\widehat{b}_i = \overline{b}_i$ for every other i.
Set $\hat{b}_i = \widehat{b}_i\ \forall i$.
Set $\hat{c}^b(e) = 0\ \forall e$; $b_i^0 = \hat{b}_i\ \forall i$; and $\ell = f = 0$.
If $2\lceil \frac{1}{2} s(N^0) \rceil < \min\{\hat{b}_i : i \in N^0 \cup N^{0,2}\}$, then set $Ind_b^f = 1$ and go to Step $4b$.
If $s(N^0) < \min\{b_{t_1}, b_{(n_1+1)}\} - 2$, then go to Step $4b$.
If $\hat{b}_{t_1} = \hat{b}_{(n_1+1)}$, then set $Index_I^b = 1$.
Else, add 2 to the smaller of $\hat{b}_{t_1}$ and $\hat{b}_{(n_1+1)}$.

Step 4b: Let $\alpha = \max\{b_i^\ell : i \in N^0 \cup N^{0,2}\}$.
If $\alpha = 0$, go to Step $5a$.
If $b_{t_1}^\ell < \alpha$, then set $x = 0$; else, let x be the largest integer such that $b_{t_x}^\ell = \alpha$.
If $b_{(n_1+1)}^\ell < \alpha$, then $y = n_1$; else, let y be the largest integer in $N^{0,2}$ such that $b_y^\ell = \alpha$.
If $x = n_0$ then set $\beta_1 = 0$; else set $\beta_1 = b_{t_{x+1}}^\ell$.
If $y = n_1$, then set $\beta_2 = 0$; else set $\beta_2 = b_{y+1}^\ell$.
Set $\beta = \max\{\beta_1, \beta_2\}$.
If the set $\{t_1, t_2, \ldots, t_x\} \cup \{n_1 + 1, n_1 + 2, \ldots, y\}$ contains only 2 elements, say $\{w, z\}$, then increase $\hat{c}^b((w, z))$ by $(\alpha - \beta)$.
Else,
if $y = n_1$, then increase the value of $\hat{c}^b(e)$ by $\frac{1}{2}(\alpha - \beta)$ for every $e \in \{(t_1, t_2), (t_2, t_3) \ldots, (t_{x-1}, t_x), (t_x, t_1)\}$;
if $x = 0$, then increase the value of $\hat{c}^b(e)$ by $\frac{1}{2}(\alpha - \beta)$ for every $e \in \{(n_1 + 1, n_1 + 2), (n_1 + 2, n_1 + 3), \ldots, (y - 1, y), (y, n_1 + 1)\}$;
otherwise, increase the value of $\hat{c}^b(e)$ by $\frac{1}{2}(\alpha - \beta)$ for every $e \in \{(t_1, t_2), \ldots, (t_{x-1}, t_x), (t_x, n_1 + 1), (n_1 + 1, n_1 + 2), \ldots, (y - 1, y), (y, t_1)\}$.

Set $b_i^{\ell+1} = \begin{cases} \beta \ \ if & i \in X \\ b_i^{\ell} \ otherwise & \end{cases}$
where $X = \{t_1, t_2, \ldots, t_x, n_1+1, n_1+2, \ldots, y.\}$ Update $f = f + (\alpha - \beta)$.
If $f \geq s(N^0)$, and $Ind_b^f = 1$, then set $\ell = \ell + 1$ and go to Step $4d$.

Step 4c: Set $\ell = \ell + 1$; go to Step $4b$.

Step 4d: Construct optimal networks $G^{b,0} = [N^0, E^{b,0}, c^{b,0}]$ and $G^{b,1} = [N^{2,0}, E^{b,1}, c^{b,1}]$, using the ModG-algorithm with $(b_i^{\ell} : i \in N^0)$ and$(b_i^{\ell} : i \in N^{0,2})$ as input vectors, respectively.
Update
$$\hat{c}^b(e) = \begin{cases} \hat{c}^b(e) + c^{b,0}(e) \ \forall e \in E^{b,0} \\ \hat{c}^b(e) + c^{b,1}(e) \ \forall e \in E^{b,1} \\ \hat{c}^b(e) \qquad\qquad otherwise \end{cases}$$

Step 5a: Set $\hat{c}(e) = \hat{c}^a(e) + \hat{c}^b(e) \ \forall e$; and $\hat{G} = [N, \hat{E}, \hat{c}]$, where $\hat{E} = \{e : \hat{c}(e) > 0\}$.
Set $\overline{c}(e) = \hat{c}(e) \ \forall e$.
If the set $Q^{b,1}$ is non-empty, then order its elements as $(k_1, k_2, \ldots, k_q)$, such that $k_1 < k_2 < \cdots < k_q$, and increase $\overline{c}(e)$ by 1 $\forall e \in \{(k_i, k_{i+\frac{q}{2}}) : i \in 1, 2, \ldots, \frac{q}{2}\}$.
If $Case = 1$, then go to Step $5b$.
Define multiset $Q^a = Q^{a,0} \cup Q^{a,1} \cup Q^{b,0}$.
If the multiset Q^a is non-empty and contains at least two distinct nodes, then order the elements of Q^a as $([1], \ldots, [p])$, in the order in which they appear in the ordered sequence $(1, 2, \ldots, n_0, n_0+1, n_0+2, \ldots, n_1)$, and increase $\overline{c}(e)$ by 1 $\forall e \in \{([i], [i + \frac{p}{2}]) : i = 1, 2, \ldots, \frac{p}{2}\}$.
Set $\overline{G} = \{N, \overline{E}, \overline{c}\}$ where $\overline{E} = \{e : \overline{c}(e) > 0\}$; and go to Step 6.

Step 5b: Define multiset $Q' = Q^{a,0} \cup Q^{b,0}$.
If the multiset Q' is non-empty and contains at least two distinct nodes, then order its elements as $([1], [2], \ldots, [p])$, in the order in which they appear in the ordered sequence $(1, 2, \ldots, n_0)$, and increase $\overline{c}(e)$ by $1 \forall e \in \{([i], [i + \frac{p}{2}]) : i = 1, 2, \ldots, \frac{p}{2}\}$.
If the set $Q^{a,1}$ is non-empty, then order its elements as $(k_1, k_2, \ldots, k_\ell)$, such that $k_1 < k_2 < \cdots < k_\ell$, and increase $\overline{c}(e)$ by 1 $\forall e \in \{(k_i, k_{i+\frac{\ell}{2}}) : i \in 1, 2, \ldots, \frac{\ell}{2}\}$.
Set $\overline{G} = \{N, \overline{E}, \overline{c}\}$ where $\overline{E} = \{e : \overline{c}(e) > 0\}$.

Step 6: Set $c^*(e) = \overline{c}(e) \ \ \forall e$.
If $Index^a = 1$, decrease $c^*((u, u+1))$ by 1.
If $Index_0^a = 1$, decrease $c^*((1, 2))$ by 1.
If $Index_0^b = 1$, decrease $c^*((t_1, t_2))$ by 1.
If $Index_1^b = 1$, decrease $c^*((n_1+1, n_1+2))$ by 1.
If $Index_I^a = 1$, then increase the value of $c^*((1, n_0 + 1))$ by 1.
If $Index_I^b = 1$, then increase the value of $c^*((t_1, n_1 + 1))$ by 1.
Set $G^* = \{N, E^*, c^*\}$, where $E^* = \{e : c^*(e) > 0\}$.

Step 7: Output the network G^* and stop.

Theorem 13 *The network G^* produced by Algorithm 5 is a feasible solution to Case 3 of the 2-INSP problem and $\sum\{c^*(e) : e \in E^*\} \leq (OPT + 3)$.*

Proof. : It is easy to see that $\sum\{c^*(e) : e \in E^*\} \leq \lceil \frac{1}{2} \sum\{(a_i + b_i) : i \in N\} \rceil + 3$.
The bound on the objective function value now follows from Lemma 7.

To prove the feasibility of the network G^*, we use the same approach as in the proof of Theorem 11. Thus, consider any cut $(X, \overline{X})$. It will be sufficient to show that $c^*[X, \overline{X}] \geq \min\{\max\{a_i : i \in X\}, \max\{a_i : i \in \overline{X}\}\} + \min\{\max\{b_i : i \in X\}, \max\{b_i : i \in \overline{X}\}\}$. Let $\min\{\max\{a_i : i \in X\}, \max\{a_i : i \in \overline{X}\}\} = a_u$; and $\min\{\max\{b_i : i \in X\}, \max\{b_i : i \in \overline{X}\}\} = b_v$. We shall consider various cases:

For convenience, let $(n_0 + 1) = j$ and $n_1 + 1 = k$.

If $u = 1$, then $a_1 \leq a_j$, $a_1 \leq \widehat{a}_1 \leq a_1 + 2$, and $(a_j - 1) \leq \widehat{a}_j \leq (a_j + 1)$.
If $\widehat{a}_1 < \widehat{a}_j$, then $\hat{a}_j \geq \hat{a}_1 \geq (a_1 + 2)$.
If $\widehat{a}_1 = \widehat{a}_j$, then $\hat{a}_j = \hat{a}_1 \geq a_1$ and $Index_I^a = 1$.
If $\widehat{a}_1 > \widehat{a}_j$, then $\hat{a}_{n_0+1} = \hat{a}_1 \geq a_1 + 1$.

Thus, in every one of these cases, using the same argument as in the proof of Theorem 11, we can show that
$\sum\{\hat{c}^a((i, j)) : i \in X; j \in \overline{X}\} \geq \hat{a}_1$.

If $u = j$, then $a_j \leq a_1$, $a_1 \leq \widehat{a}_1 \leq a_1 + 2$, and $(a_j - 1) \leq \widehat{a}_j \leq (a_j + 1)$.
If $\widehat{a}_j < \widehat{a}_1$, then $\hat{a}_1 \geq \hat{a}_j \geq (a_j + 1)$.
If $\widehat{a}_1 = \widehat{a}_j$, then $\hat{a}_1 = \hat{a}_j \geq a_j + 1$ and $Index_I^a = 1$.
The case $\widehat{a}_j > \widehat{a}_1$ is not possible.

Thus, in every one of these cases, using the same argument as in the proof of Theorem 11, we can show that
$\sum\{\hat{c}^a((i,j)) : i \in X; j \in \overline{X}\} \geq \hat{a}_{j_1}$.

If $u \in N^0 - \{1\}$, then $a_u \leq a_1$, and $\hat{a}_1 \geq \hat{a}_u \geq a_u$ and if $\hat{a}_u = a_u$, then $u \in Q^{a,0}$.

In this case, using the same argument as in the proof of Theorem 11, it follows that
$\sum\{\hat{c}^a((i,j)) : i \in X; j \in \overline{X}\} \geq \hat{a}_u$.

If $u \in N^{2,0} - \{j\}$, then $a_u \leq a_j$, and $\hat{a}_j \geq \hat{a}_u \geq a_u$ and if $\hat{a}_u = a_u$, then $u \in Q^{a,1}$.

In this case too, using the same argument as in the proof of Theorem 11, it follows that
$\sum\{\hat{c}^a((i,j)) : i \in X; j \in \overline{X}\} \geq \hat{a}_u$.

It can be similarly shown that
$\sum\{\hat{c}^b((i,j)) : i \in X; j \in \overline{X}\} \geq \hat{b}_v$.

If nodes 1 and j (t_1 and k) lie on different sides of the bipartition $(X, \overline{X})$ then if possible, choose $u \in \{1, j\}$ ($v \in \{t_1, k\}$), preferably $u = j$ ($v = t_1$).

Then, using the same approach as in the proof of Theorem 11, the above facts and the choice of sets $Q^{a,0}$, $Q^{a,1}$, $Q^{b,0}$, $Q^{b,1}$, it can be easily shown that $c^*[X, \overline{X}] \geq \overline{a}_u + \overline{b}_v$.

Except for the case $(u \in N^1, v \in N^0)$, the result now follows from the fact that $\overline{a}_u + \overline{b}_v = a_u + b_v$.

Let us consider the case $u \in N^1, v \in N^0$.

If $u = j$ or $v = t_1$, then using the same approach it can be easily shown that $c^*[X, \overline{X}] \geq \overline{a}_u + \overline{b}_v + 1 = a_u + b_v$.

Now let us consider the case $u \in N^1 - \{j\}\}, v \in N^0 - \{t_1\}$.

If nodes 1 and j lie on different sides of the bipartition $(X, \overline{X})$, then it follows by the choice of the node u that IndexaI=0. Similarly, if nodes t_1 and k lie on different sides of the bipartition $(X, \overline{X})$, then IndexbI=0. Thus, in either case, $c^*[X, \overline{X}] \geq \overline{a}_u + \overline{b}_v + 2 > a_x + b_y$.

Else, each of node sets N^0 and N^1 intersects properly each of the sets X and $\overline{X}$. Thus, if we traverse the node set $N^0 \cup N^1$ in the order $((j =)n_0 + 1, n_0 + 2, \ldots, n_1, 1, 2, \ldots, n_0, n_0 + 1)$, then we cross the cut $(X, \overline{X})$ at least 4 times. Using the same argument as in the proof of Theorem 11, it now follows that $c^*[X, \overline{X}] \geq \overline{a}_u + \overline{b}_v + 1 = a_u + b_v$. This proves the result. ■

:

5. The General Case

We shall now use the insights gained from algorithms for the special cases presented in Section 4. to develop an algorithm for the general case that produces a feasible network with objective function value within 3 of the lower bound established in Section 3..

Step 0: Compute $a_i = \max\{r_{ij} : j \neq i\}$ and $b_i = \max\{s_{ij} : j \neq i\}$ $\forall i \in N$.

Let $\overline{N} = N^{0,2} \cup N^{1,2} \cup N^{2,0} \cup N^{2,1} \cup N^{1,1} \cup N^{2,2}$.

Design network $\overline{G} = \{\overline{N}, \overline{E}, \overline{c}\}$, for input $\{a_i, b_i : i \in \overline{N}\}$ using Algorithm 5 of case 3 of the previous section.

Let $c^*(e) = \overline{c}(e)$ $\forall e \in \overline{E}$.

Let $\tilde{N} = N^{0,1} \cup N^{1,0} = N - \overline{N}$. Let $\tilde{G} = [N, \tilde{E}]$ where $\tilde{E} = \{(i,j) : r_{ij} > 0 \text{ or } s_{ij} > 0\}$. Contract in $\tilde{G}$ the node set $\overline{N}$ to get graph G'. Find a spanning forest in G'; and assign $c^*(e) = 1$ for all edges e in G' corresponding to this spanning forest. Let $E^* = \{e : c^*(e) > 0\}$. Output network $G^* = [N, E^*, c^*]$ and stop.

Theorem 14 *The network G^*, output by Algorithm 5., is feasible to the problem 2-INSP and has objective function value no more than $(OPT + 3)$.*

Proof. This follows easily from Theorem 13 and Lemma 6. ■

References

[1] Y. P. Aneja, R. Chandrasekaran, S. N. Kabadi and K. P. K. Nair. Flows over Edge-Disjoint Mixed Multipaths and Applications, *Discrete Applied Mathematics*, 155(15), 1979-2000, 2007.

[2] R. Chandrasekaran, K.P.K. Nair, Y.P. Aneja and S.N. Kabadi. Multi-terminal Multipath Flows: Synthesis, *Discrete Applied Mathematics*, 143, 182-193, 2004.

[3] W. Chow and H. Frank. Survivable Communication Networks and the Terminal Capacity Matrix, *IEEE Trans. on Circuit Theory*, CT-17, 2, 192-197, 1970.

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

[5] R. J. Gibbens and F. P. Kelly. Dynamic Routing in Fully Connected Networks, *IMA Journal of Mathematical Control and Information*, 7, 77-111, 1990.

[6] D. Gusfield. Simple Constructions for the Multi-Terminal Network Flow Synthesis, *SIAM J. Comput.*, 12, 1, 157-165, 1983.

[7] R.E. Gomory and T.C. Hu. Multi-terminal network flows, *Journal of SIAM*, 9, 551-570, 1961.

[8] R. Hassin and A. Levin. Synthesis of 2-Commodity Flow Networks, *Mathematics of Operations Research*, 29 (2), 280-88, 2004.

[9] T. C. Hu. Multi-Commodity Network Flows, *Operations Research*, 11, 344-360, 1963.

[10] A. Itai, Y. Perl and Y. Shiloach. The Complexity of Finding Maximum Disjoint Paths with Length Constraints, *Networks*, 12, 277-286, 1982.

[11] S. N. Kabadi, R. Chandrasekaran, K.P.K. Nair and Y. P. Aneja. Integer Version of the Multipath Flow Network Synthesis Problem, *Discrete Applied Mathematics*, 156(18), 3376-99, 2008.

[12] S. N. Kabadi, J. Kang, R. Chandrasekaran and K. P. K. Nair. Hop-Constrained Network Flows: Analysis and Synthesis, *working paper, Faculty of Business Administration, University of New Brunswick*, October, 2003.

[13] S. N. Kabadi and K. P. K. Nair. Integer Network Synthesis Problem for Hop Constrained Flows, *working paper, Faculty of Business Administration, University of New Brunswick*, June 2009.

Received 2-1-2009; revised 26-6-2009; accepted 2-7-2009

[14] S. N. Kabadi and R. Sridhar. Peeling Algorithm for integral Network Synthesis, *working paper, Faculty of Business Administration, University of New Brunswick*, January, 1996.

[15] W. Kishimoto. A method for obtaining the maximum multi-route flows in a network, *Networks*, 27, 279-291, 1996.

[16] W. Kishimoto and M. Takeuchi. On m-route flows in a network, *IEICE Trans.* J-76-A , 1185-1200, 1993 (in Japanese).

[17] W. Mayeda. Terminal and branch capacity matrices of a communication Net, *IRE Transactions on Circuit Theory*, CT-7, 261-269, 1960.

[18] S. T. McCormick. The Complexity of Max Flow and Min Cut with Bounded-Length Paths, *working paper, Faculty of Commerce and Business Administration, University of British Columbia*, May, 2001.

[19] A. Schrijver, Combinatorial Optimization: Polyhedra and Efficiency, Algorithms and Combinatorics 24, Springer-Verlag, New York, 2003.

[20] R. Sridhar and R. Chandrasekaran. Integer Solution of Synthesis of Communication Network. *Mathematics of Operations Research*, 17, 3, 581-585, 1992.

[21] K. Talluri. Network Synthesis with Few Edges, *Networks*, 27, 109-115, 1996.