Bulletin of the *Transilvania* University of Braşov • Vol 13(62), No. 1 - 2020
Series III: Mathematics, Informatics, Physics, 303-330
https://doi.org/10.31926/but.mif.2020.13.62.1.23

MODIFIED PULAT'S ALGORITHM FOR THE MAXIMUM OUTFLOW PROBLEM IN UNDIRECTED GENERALIZED NETWORKS

Massoud AMAN[1], Reza GHANBARI[2] and Donya HEYDARI[*,3]

Abstract

A generalized network is characterized by arc multipliers that specify which portion of the flow entering an arc at its tail node reaches its head node. In this paper, the goal is to maximize the flow excess at the sink in undirected generalized networks. We show that the results of the directed networks are not satisfied here. We have specifically stated and proved the optimality conditions of the problem. We have also designed an algorithm according to Pulat's algorithm and achieved an efficient version.

2000 *Mathematics Subject Classification:* 05C21, 05C82
Key words: maximum outflow problem, generalized networks, undirected networks.

1 Introduction

In generalized networks, there is a factor named multiplier defined on each arc that is positive. Arc multipliers are not all equal to one. Indeed, they denote the amount of flow reached the tail node if we enter one unit of flow to the arc. This factor makes generalized networks distinctly different from traditional networks (see Ahuja [2]). In this paper, we consider *undirected* networks. It is given in [2] Appendix B and [27] that the undirected maximum flow problem in traditional networks with nonenegative lower bounds is NP-hard.

In generalized networks, gains (when the arc multiplier is greater than one) and losses (when the arc multiplier is less than one) can refer to evaporation,

[1]Department of Mathematics, Faculty of Mathematics and Statistics, University of Birjand, Birjand, Iran, e-mail: mamann@birjand.ac.ir
[2]Department of Applied Mathematics, Faculty of Mathematical Sciences, Ferdowsi University of Mashhad, Mashhad, Iran, e-mail: rghanbari@um.ac.ir
[3]* *Corresponding author*, Department of Mathematics, Faculty of Mathematics and Statistics, University of Birjand, Birjand, Iran, e-mail: do_heydari@birjand.ac.ir

energy dissipation, breeding, theft, interest rates, blending, currency exchange, and transforming one commodity into another (see Ahuja [2] and [47]).

Here, we study the generalized flow maximization problem in undirected networks, where the objective is to send the maximum amount of flow to the sink. This problem is named generalized maximum outflow problem in undirected networks (GMOPUN).

The generalized network problems have several applications in operations research [Ahuja [2], Chapter 15], electrical systems (e.g. Adams et al. [1], Sasson [40], Taluktar and Morton [42], and Liu and Wu [33]), financial problems (e.g. Crum et al. [9], Guim and Nye [24], and Robichek, Teichroew and Jones [39]), water distribution systems (e.g. Bhaumik [6]), copper refining process (e.g. Kim [31]), air force course scheduling problem (e.g. Glover et al. [16]) and lot sizing problems (e.g. Steinberg and Napier [41]). Additional applications of the generalized network flow problem are in resort development (e.g. Glover and Rogozinski [19]), airline seat allocation problems (e.g. Glover et al. [17], and Dror, Trudeau and Ladany [11]), personnel planning (e.g. Gorham [23]), a consensus ranking model (e.g. Barzilai, Cook and Kress [5]), cash flow management in an insurance company (e.g. Crum and Nye [10]), and land management (e.g. Glover, Glover, and Martinson [15]). Glover et al [16], and Glover, Klingman and Phillips [18] contain additional references concerning applications of generalized network flow problems.

The GMOPUN is motivated by real-world and a wide range of theoretical and practical applications of undirected networks. Consider transporting raw materials to provide the demands of some factories and research and investigation departments in which commodities and materials are oxidized ,or combine with each other, evaporate and dissipated without considering the direct of the transport routes. We also can take transporting medicinal substances to hospitals and clinics or some activities such as blood transfusion as some notable examples of applications.

Since the early 1960s, there has been substantial amount of research done on directed generalized networks. We can cite the work of Charnes and Cooper [8], Eiseman [12], Balas and Ivanescu [4], and Balas [3] for the generalized transportation problems.

The first combinatorial algorithms for the directed generalized maximum flow problem were exponential-time augmenting path algorithms proposed by Jewell [30] and Onaga [35]. Truemper [43] observed that by using the cost function $c = log\gamma$ where γ is the arc multiplier, many of the early generalized maximum flow algorithms were, in fact, analogue of pseudo-polynomial minimum cost flow algorithms. Jewell's [30] generalized maximum flow algorithm is an analogue of Klein's [32] cycle cancelling algorithm for the minimum cost flow problem. Similarly, Onaga's [35] algorithm is analogous to the successive shortest path algorithm developed independently by Busacker and Gowen [7], Iri [26], and Jewell [29]. Truemper's [44] algorithm is an analogue of Ford and Fulkerson's [13] primal-dual algorithm. Jarvis and Jezior's [28] algorithm is an analogue of Fulkerson's

[14] out-of-kilter algorithm. Goldberg, Plotkin, and Tardos [20] designed the first two polynomial-time combinatorial algorithms for the generalized maximum flow problem. Radzik [37] modified the Fat-Path algorithm and then Goldfarb and Jin [21] designed an algorithm based on the minimum cost flow algorithm. Wayne [47] developed the first efficient primal algorithm for the problem. Restrepo and Williamson [38] designed an algorithm based on cancelling generalized augmenting paths. In 2016 and 2017, Vegh [45] and Olver and Vegh [34] designed two strongly polynomial algorithms for generalized flow maximization. There is another approach based on cancelling particular paths and cycles in directed generalized networks constructed by Pulat [36] for maximum out flow problem. Indeed, Pulat's algorithm has increased the excess of the sink by, first augmenting flow through the s-t paths in a maximal acyclic subnetwork and then determining the maximum out flow in the rest of the network.

We are not able to find any algorithms designed particularly for undirected generalized network problems. It is hard to handle network problems in undirected generalized networks (the necessity of the problem with details is given in next paragraph and subsequent sections).

In network theory, in order to apply the algorithms of directed networks for undirected ones, it is essential to convert undirected networks to directed ones mostly by replacing an edge with two directed arcs. Due to the existence of the arc multipliers, not only this common conversion will create some significant problems and ambiguity in residual network, but also we can not use usual definitions such as cycles, residual capacity and flow augmentation (see Ahuja [2]). So we redefine some essential notations and define a particular structure of residual network.

In directed generalized networks, just one type of cycle increases the amount of flow at a sink node. But, in directed generalized networks constructed from undirected generalized networks, all types of cycles in some cases could create excess at the sink node. Consequently, the common optimality condition ([20], [22]) of generalized maximum outflow problem in directed networks is not satisfied here. Then achieving the optimal solution is a complicated task which has made our proposed algorithm entirely different from Pulat's algorithm. In next section, we have stated and proved the optimality condition of our problem.

In Section 2, we provide some definitions and introduce some significant terminology. The GMOPUN is also stated. In Section 3, we name our proposed algorithm "modified Pulat's algorithm (MPA)". Section 4 is devoted to the study of one sub-function of the MPA and its influence on the efficiency of MPA. Section 5 includes an example. We conclude in Section 6.

2 Terminology and Modelling

We consider an undirected generalized network $G = (N, E)$ where N is the set of nodes and E is the set of edges, with $|N| = n$ and $|E| = m$. Let s and t denote the source and sink nodes, respectively.

Suppose $\{i, j\}$ shows an edge between two nodes i and j in E. if $\{i, j\}$ origi-

nates from i to j or in reverse direction, it is written as (i,j) or (j,i), respectively. Associated with each $\{i,j\}$, there are two non-negative parameters $u\{i,j\}$, the upper bound of the flow that can enter into the edge, and $\gamma\{i,j\}$, the edge multiplier. We have $u_{ij} = u_{ji} = u\{i,j\}$ and $\gamma_{ij} = \gamma_{ji} = \gamma\{i,j\}$. If some flow sent from i to j or j to i, is shown by f_{ij} or f_{ji}, respectively. If f_{ij} units of flow are sent from i to j (j to i), $\gamma_{ij}f_{ij}$ ($\gamma_{ji}f_{ji}$) units will be received in j (i).

A *generalized pseudoflow* is a function $f : E \longrightarrow \mathbb{R}$ which for every edge satisfies capacity constraints $f\{i,j\} \leq u\{i,j\}$ and anti symmetric constraints $f_{ji} = -\gamma_{ij}f_{ij}$.

A *generalized flow* is a generalized pseudoflow such that the conservation of flow (2) is held at all nodes except s and t.

A *blocking flow* from s to t is a generalized flow that has saturated at least one arc on each path from s to t. We name the saturated arcs *blocking arcs*.

In undirected generalized networks, an edge can be used in both directions, we consider here that if it is used in one way, it can not be used in the other direction, i.e., $f_{ij}f_{ji} = 0$.

Converting the undirected generalized networks to the directed ones is necessary for the MPA. So, each edge $\{i,j\}$ is replaced by two arcs (i,j) and (j,i). For notational convenience, the new directed network is denoted by $\mathbf{G} = (N, \mathbf{A})$ and named *constructed* directed network. We define residual network $\mathbf{G}^r = (N, \mathbf{A}^r)$. The residual network consists of only the arcs with a positive residual capacity (it is defined in Definition 1).

Consider generalized flow f in $\mathbf{G}$. The GMOPUN in constructed directed network is;

$$Maximize \quad \Sigma_{\{j:(j,t)\in\mathbf{A}\}}\gamma_{jt}f_{jt} \tag{1}$$

Subject to

$$\Sigma_{\{j:(i,j)\in\mathbf{A}\}}f_{ij} - \Sigma_{\{j:(j,i)\in\mathbf{A}\}}\gamma_{ji}f_{ji} = 0 \qquad \forall i \in N - \{s,t\} \tag{2}$$

$$0 \leq f_{ij} \leq \alpha_{ij}u_{ij} \qquad \forall (i,j) \in \mathbf{A} \tag{3}$$

$$0 \leq f_{ji} \leq (1-\alpha_{ij})u_{ji} \qquad \forall (j,i) \in \mathbf{A} \tag{4}$$

$$\alpha_{ij} \in \{0,1\} \qquad \forall (i,j) \in \mathbf{A} \tag{5}$$

The capacity constraints are (3) and (4). In fact, we need to use α_{ij} because of the constraint $f_{ij}f_{ji} = 0$.

Take Figure 1 in $\mathbf{G}^r$. Let f_{ij} units of flow be sent from i to j in solid line. Then there exist $\gamma_{ij}f_{ij}$ units of flow in j and immediately the reverse arc shown by dotted line in Figure 1 will appear by arc multiplier $\frac{1}{\gamma_{ij}}$. Now, if some flow should be sent from j to i, first the flow (maximum $\gamma_{ij}f_{ij}$ units) must be sent along the dotted arc and then, if necessary, along the solid arc from j. Because the available flow in j should be sent back first and then sending extra flow is allowable. Since the arc multipliers of solid and dotted arcs are different, we cannot consider them as one arc and consequently we need Definition 1.

Now let some flow be sent through a cycle $i - j - i$ (solid lines) in Figure 1. Some excess or deficit or none is created depending on the arc multipliers. This

excess or deficit is artificial (unreal) and also since just one direction of an edge can be used, then these cycles must not be detected. Definition 1 can help us not to detect them. The word "cycle" in this paper excludes $i - j - i$.

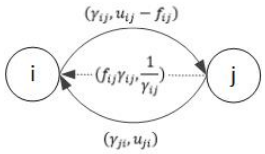

Figure 1: The dotted and solid lines show the cancel and real arcs, respectively

Definition 1. *Let f be a generalized flow. If f_{ij} units of flow are sent from i to j, the reverse arc of (i, j) is added to the residual network of the constructed directed network. Arc (i, j) is called real arc with updated residual capacity $u^r_{ij} = u_{ij} - f_{ij}$. Moreover, the reverse arc (j, i) is named cancel arc and to distinguish from real arc (j, i), it is shown by $(\overline{j, i})$. Its residual capacity and arc multiplier are respectively $u^r_{\overline{ji}} = \gamma_{ij} f_{ij}$ and $\gamma_{\overline{ji}} = \frac{1}{\gamma_{ij}}$.*

If both arcs, real and cancel like (i, j) and $(\overline{i, j})$ exist in $\mathbf{G}^r$, we name them *parallel* arcs. Note that there could be at last 3 arcs between two nodes in $\mathbf{G}^r$ (Figure 1 shows an example.). If both real arcs (i, j) and (j, i) exist in $\mathbf{G}^r$, we name them *active* arcs and we name nodes i and j *active* nodes. If we do not mention the type of the arc and just write arc (i, j), we mean both types real and cancel.

An augmenting path (AP) is a residual s-t path. The product of all arc multipliers, that is called gain factor, on an AP (or cycle) like P is given by $g(P) = \Pi_{(i,j)\in P}\gamma_{ij}$. There are three types of cycles on generalized networks according to their gain factors; flow generating cycle (FGC) (when $g(C) > 1$), unit gain cycle (UGC) (when $g(C) = 1$) and flow absorbing cycle (FAC) (when $g(C) < 1$). Bicycle is an FGC, an FAC cycle and a (possibly trivial) path connecting the two cycles. A generalized augmenting path (GAP) is a residual FGC, together with a (possibly trivial) residual path from a node on the cycle to the sink.

Remark 1. *To detect a residual cycle or an AP when we follow nodes, if we meet a node with parallel arcs, the cancel arc should be selected as an arc of the cycle or AP.*

To know the type of a residual cycle like C_1, $g(C_1)$ should be calculated first. Take Figure 2 as an illustration. According to remark 1, the detected residual cycle C_1 (the left shape) in Figure 2 is $1 - \overline{2 - 3} - 1$. Suppose some flow is augmented in C_1 and let arc $(\overline{2, 3})$ be saturated. Afterwards, there exists another residual cycle named C_2 (the right shape) in Figure 2, $1 - 2 - 3 - 1$ with the same set of nodes but a different type. C_1 is the FAC ($g(C_1) = (1/2)(1/2)(1/5)$) but C_2 is the FGC ($g(C_2) = (1/2)(1/2)(5)$). In fact, the introduction of $(2, 3)$ to the network creates another cycle with a different type which was hidden before and after augmentation of C_1 appeared.

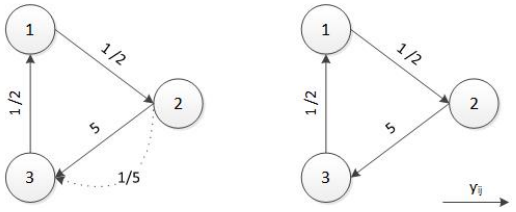

Figure 2: An example of an FAC and an FGC.

In directed generalized networks, since there are no parallel arcs, after augmenting a cycle, there will not be another cycle with the same sets of nodes. Consequently, FGCs are the only cycles that create excess at t. However, in undirected generalized networks all types of cycles could create some excess at t since they might make an FGC be hidden. Then all types of residual cycles should be detected and checked if after their augmentation, cycles will be created (one after each other).

The following definitions and paragraphs help us deal with all types of residual cycles and describe the importance of them

Definition 2. *An AP between two distinct nodes or a residual cycle with parallel arcs is named multiple AP or multiple residual cycle, respectively. Otherwise it is named simple.*

Obviously, a simple AP or residual cycle is a kind of multiple one. Take Figure 2 (the left shape) and 3 as an illustration of multiple residual cycle and AP, respectively.

Figure 3: An $s-t$ multiple AP

Definition 3. *Consider P as a multiple AP (multiple residual cycle) in $\mathbf{G}^r$. The first sub-augmenting path (sub-cycle) that is detected according to Remark 1 is named basis.*

$1-\overline{2-3}-1$ and $\overline{s-1-2}-\overline{3-t}$ are the basis in Figure 2 (the left shape) and 3, respectively.

Definition 4. *Given a residual multiple cycle with a residual UGC or FAC as its basis together with an (possibly trivia) AP connecting the residual multiple cycle to t. If the excess of t is increased by augmenting flow through the basis and some of its subsequent subcycles, the residual multiple cycle and its AP is called multiple generalized augmenting path (MGAP).*

Definition 5. *A subnetwork of* $\mathbf{G}^r$ *is named a basic network if it consists of only the simple APs, simple cycles and the bases of the multiple APs and cycles. We show the basic network by* $\mathbf{G}^b = (N, \mathbf{A}^b)$.

The APs and residual cycles of $\mathbf{G}^r$ allowed to be augmented are all in $\mathbf{G}^b$. Then, we can detect them easily in $\mathbf{G}^b$.

Given a generalized pseudoflow f in a *directed* generalized network. f is associated with a subnetwork induced by the arcs on which f is positive.

The structures of the subnetwork are: a path between two distinct nodes (it creates deficit at the first node and excess at the end node), a UGC (it does not create any excess or deficit), an FGC and a path connecting the cycle to a node (it creates excess at the end of the path), an FAC and a path connecting the cycle to a node (it creates deficit at the end of the path), a bicycle (it does not create any excess or deficit). The five structures are named *elementary pseudoflows*. According to the following theorem from [20] and [22], a generalized pseudoflow in a directed generalized network can be decomposed into elementary pseudoflows.

Theorem 1. *Given a generalized pseudoflow f. It can be decomposed into components* $f_1, f_2, ..., f_k$ *with* $k \leq m$ *such that*

$$f(i,j) = \sum_l f_l \tag{6}$$

Now consider G and its corresponding constructed directed network $\mathbf{G}$. All five elementary pseudoflows and Theorem 1 are satisfied in $\mathbf{G}^b$ as well as directed generalized networks.

We claim that s-t APs, GAPs and MGAPs are the only three ways to generate excess at t in $\mathbf{G}^r$. The following theorem proves our claim.

Theorem 2. *A generalized flow in* $\mathbf{G}$ *is a generalized maximum outflow if and only if* $\mathbf{G}^r$ *has no* $s-t$ *APs, GAPs or MGAPs.*

Proof. Clearly if a generalized flow f has an s-t AP, GAP or MGAP in $\mathbf{G}^r$ then it is not optimal. Now, suppose f has no s-t Aps, GAPs or MGAPs in $\mathbf{G}^r$. Let f^* be a generalized maximum outflow. Decompose $f^* - f$ in $\mathbf{G}^b$ according to Theorem 1. Then use the decomposition in $\mathbf{G}^r$. The set of the elementary pseudoflows denotes how to get f^* from f. If we show there is none in the decomposition to create excess at t, f is optimal. There are five cases according to elementary pseudoflows as follows;

Case 1 : The decomposition has no s-t APs. Consequently, if t belongs to a residual path, it does not have a flow generator (source node) or if s belongs to a residual path, augmenting flow in the path cannot create any excess at t.

Case 2 : If t belongs to a residual UGC, by augmenting flow through the residual UGC the excess of t will not change. Therefore, a residual UGC can create excess at t only if it is a basis of an MGAP. But it is impossible since f has no MGAP.

Case 3 : A residual FGC and a (possibly trivia) path in the decomposition creates excess at t only if t is the endpoint of the path. Then there is a GAP. But it is a contradiction since f has no GAP.

Case 4 : If t belongs to a residual FAC and a (possibly trivia) path, by augmenting flow through it the excess of t is decreased. Therefore, it can create excess at t only if it is a basis of an MGAP. But it is impossible since f has no MGAP.

Case 5 : t cannot belong to any residual bicycles. Otherwise there is a GAP and it is a contradiction. Then the excess of t could not change by augmenting flow through any residual bicycles.

□

Let us introduce some further notation. A *maximal acyclic* subnetwork of $\mathbf{G}^r$ is an acyclic subnetwork $G_1 \subset \mathbf{G}^r$ such that there exists no other acyclic subnetwork of $\mathbf{G}^r$ that contains G_1. Clearly, it is not unique. In this paper, due to the possible existence of closed paths including two real arcs like (i, j) and (j, i) in G_1, we name the maximal acyclic subnetwork G_1, *maximal acyclic real subnetwork*. Although here these closed paths are not considered as cycles, this name is chosen for G_1 to be distinguishable from usual maximal acyclic subnetwork. Let G_2 denote the subnetwork of G_1^b formed by all $s-t$ APs with maximum gain factor. Let G_3 denote the subnetwork of G_2 which contains all the $s-t$ APs with minimum numbers of arcs.

3 Our proposed algorithm

The principle of the algorithm is to saturate (augmenting some flow to saturate at least one arc of an AP or GAP) all the $s-t$ APs before GAPs . Then detecting and saturating the MGAPs. It has five main steps, with each step executed at most once.

Algorithm 1: The modified Pulat's algorithm
Input: an undirected generalized network.
Output: maximum outflow

1. Construct a directed network $\mathbf{G}$.
2. Construct a maximal acyclic real subnetwork G_1 of $\mathbf{G}^r$.
3. Augmenting flow in $s-t$ APs in G_1.
4. Augment (extend) G_1 with original arcs not in G_1, one at a time, and augmenting flow through the resulting $s-t$ APs, if any and then in GAPs.
5. Determining the maximum out flow in $G_1 \bigcup G_1^c$.

In subsequent subsections, Steps 2, 3, 4 and 5 are described in details.

3.1 Construct a maximal acyclic real subnetwork

This part includes two steps. Initial construction of $G_1 = (N_1, A_1)$ (for simplicity notation, A_1^r is denoted by A_1) and a step to extend G_1 together with checking for cycles.

The first step is the breadth-first-search (BFS) method (see Horowitz and Sahni [25]). We start with node s in $\mathbf{G}^r$ which has BFS number zero. All the nodes that can be reached from node s will have a BFS number one. Node s is then said to be explored. In general, an unexplored node with BFS number l is picked, all nodes without BFS numbers which can be reached from this node are assigned BFS number $l+1$. BFS number to node t is assigned last. All arcs joining nodes with a BFS number of the predecessor node less than the BFS number of the successor node are put in A_1. The remaining arcs are put in ${A_1}^c$.

After applying BFS method, the resulting network is always a directed-out tree since if both arcs (i,j) and (j,i) exist in $\mathbf{G}^r$, one is chosen depending on whether $l_i < l_j$ or $l_i > l_j$ and clearly the selected arc lies on a path from s to other nodes.

The next step tries to add arcs in ${A_1}^c$ to A_1 one at a time checking whether or not the new arc forms a cycle with the network constructed so far.

Let (i,j) be one arc in ${A_1}^c$. Then, (i,j) can be just real since we have not augmented flow in G_1^c yet. Consequently, there are only two cases;

Firstly, if real arc (j,i) is not in A_1, adding and checking can be accomplished by using the fact that each node in a cycle is a predecessor node of one arc and a successor node of another arc. One can iteratively eliminate nodes (along with the incident arcs) of the network which do not satisfy the above condition. If the network is emptied, then the new arc does not form a cycle with the arcs in A_1, so it is added to A_1. Otherwise, it is kept in ${A_1}^c$.

Secondly, if real arc (j,i) is in A_1, to avoid detecting $i-j-i$, first (j,i) is eliminated from A_1 and afterwards we carry out the same procedure in the preceding paragraph. Finally, (j,i) should be added to A_1 again.

3.2 Augment flow in $s-t$ APs in G_1.

There are six main steps.

First we construct subnetwork G_2 of G_1^b which includes only $s-t$ APs with the maximum gain $g(.)$. Obviously, finding a path with maximum gain $g(.)$ is equivalent to finding a path with minimum cost with arc cost $c_{ij} = -log(\gamma_{ij})$. G_2 can be constructed in O(mn) time by BellmanFord shortest-path algorithm (If all the edge multipliers of G are less than or equal to one, G_2 can be constructed in O(m) time by Dijkstras shortest-paths algorithm). To use Bellman-Ford algorithm, the network should not contain any negative cost cycles. Therefore, if there are active nodes like i and j in G_1^b with arc multiplier greater than one, we should choose one of the active arcs. So, when active node i is selected to be scanned for the first time in first iteration of Bellman-Ford algorithm, if $d(j) \neq \infty$, then we set $\tau = \min(d(j) + c_{ij}, d(i) + c_{ji})$. If active node i (j) is made τ, we delete arc (i,j) $((j,i))$ temporarily from G_1^b and when Bellman-Ford algorithm is terminated we add it again to G_1^b.

There might be more than one s-t AP with maximum gain, in this case the layered subnetwork G_3 of G_2 contains all $s-t$ APs in G_2 with minimum number of arcs. This generation of G_3 has been done in Proc *Layer* given in Pulat [36].

We design Proc *Advanced Block flow* given with details in Section 4 to calculate the blocking flow. When there is a multiple $s-t$ AP (a residual cycle), there are two ways to augment flow in them. First, to augment only through its basis. Second, to augment flow through all multiple $s-t$ AP (residual cycle), if it is possible (Proposition 2). Both ways can be done by Proc *Advanced Block flow*. The comparison of two ways is also given in Section 4.

Algorithm 2: Proc Outflow in G_1
Input: G_1.
Output: maximum outflow in G_1.

1. Use Bellman-Ford algorithm to generate the maximum gain subnetwork $G_2 \subset G_1^b$ from s to t.

   (1.1) **If** no path can be found, **then** go to Step 6; **else** go to Step 2.

2. Use Proc *Layer* to determine the layered network $G_3 \subset G_2$.

   (2.1) **If** no layered network can be found, **then** go to Step 1; **else** go to Step 3.

3. Use Proc *Advanced Block flow*.

4. $R_1 = \{(i,j) \in G_1 : (\overrightarrow{i,j}) \in s-t \text{ AP}\}$.

   (4.1) **For** all $(i,j) \in R_1$ **if** $\overline{u}_{ij} = 0$ and $\gamma_{ij} > 1$, **then** $G_1 = G_1 - \{(i,j)\}$ and $G_1^c = G_1^c \bigcup (i,j)$.

5. Use Proc *Update Cap* by blocking flow (BF) to update arc capacities in G_2 and G_1.

   (5.1) F=F+ BF and go to Step 1.

6. The outflow from s to t in G_1, $\overline{v_t}$, is given by $\overline{v_t} = \Sigma_{i \in B(t)} \gamma_{it} f_{it}$.($B(t)$ is the set of immediate predecessor nodes to t.)

Although G_1 is referred to as an acyclic real network, it is acyclic only if the original arcs are considered. In fact, after the first augmentation in subnetwork G_1, the reverse arcs might create some FGCs. Proposition 1 and 2 indicate that if the $s-t$ AP is simple, the created cycles will not be FGCs and if the $s-t$ AP is multiple, under what circumstances the created cycles will not be FGCs, respectively. So, for a multiple $s-t$ AP that does not satisfy Inequality 7, we augment flow only through its basis.

Proposition 1. *The capacity update routine (Proc Update Cap) will not create FGCs in G_1 after augmenting flow in the basis of a multiple $s-t$ AP or simple $s-t$ AP.*

Proof. This is proved similar to Proposition 1 in Pulat [36] by considering remark 1 for GMOPUN. □

Proposition 2. *Consider P as a multiple $s-t$ AP with a maximum gain basis and let R show the set of all used (augmented) real arcs among parallel ones after the augmentation of P. If P satisfies the following property*

$$g(R) = \prod_{(i,j) \in R} \gamma_{ij} \geq 1. \tag{7}$$

after capacity alteration, no FGCs will be created by the reverse arcs.

Proof. Let P_1 be the basis of P with the maximum gain factor among all available bases. There are two cases;
First, if $R = \varnothing$, it is proved as same as proposition 1.
Second, if $R \neq \varnothing$, let H show the set of parallel arcs of P. Then, we have the following sets;

$$F_1 = \{(\overline{i,j}) : (i,j) \in H - R\}$$

$$F_2 = \{(\overline{i,j}) : (i,j) \in R\}, \quad E = P - H$$

Flow increase along path P causes an increase in the capacity of the reverse arcs. Let P' show the reverse multiple AP. We should show that the basis of P', P'_1, will not form an FGC (arcs of P'_1 are the reverse of R, F_1 and E). Recall that P_1 is the maximum gain basis in G_2. Therefore, any $s-t$ AP like P_2 must have a gain factor $g(P_2) \leq g(P_1)$. Then

$$g(P_2) \leq g(P_1) = g(E)g(F_1)g(F_2) \tag{8}$$

Also, we have

$$g(R) = \frac{1}{g(F_2)} \tag{9}$$

If P_1' and P_2 create a cycle like C, its gain factor is not greater then one since

$$g(C) = g(E')g(F_1')g(R')g(P_2) = \frac{g(P_2)}{g(E)g(F_1)g(R)} \tag{10}$$

According to 7, 8 and 9 we have

$$g(C) \leq g(F_2)^2 \leq 1 \tag{11}$$

□

If we augment flow through the $s-t$ multiple APs or only their bases and some cancel arcs among parallel arcs are saturated, their corresponding real arcs will appear (we call them *appeared real arcs*) in G_1. The appeared real arcs might create FGCs. Therefore, Step 4 of Algorithm 2 is given to avoid the existence of the created FGCs in G_1. There are two cases:

First, if the gain factors of the appeared real arcs are less than or equal to one, then there will not clearly be any FGCs (because the gain factors of their corresponding cancel arcs are greater than one and they were not part of any FGCs before. So the appeared real arcs are not part of any FGCs.)

Second, if the gain factors of the appeared real arcs are greater than one, then they might be part of an FGC. So, we eliminate them from G_1 and add them to G_1^c to be checked later one by one whether they create any FGCs.

The following proposition proves the finiteness of the algorithm.

Proposition 3. *No AP is used more than once.*

Proof. This is proved similar to Proposition 2 in Pulat [36] by considering remark 1 for GMOPUN. □

3.3 Augmenting flow in augmented G_1 through $s-t$ APs and GAPs

The introduction of any arc from A_1^c into G_1 might create a cycle which may be an FGC and as a result a GAP, and may also create an $s-t$ AP in G_1. We show the augmented G_1 (when an arc is added to G_1) by $G_1' = (N_1', A_1')$. Since the principle of our method is to saturate all the APs before GAPs, dealing with APs created by introducing new arc from A_1^c should be reached instantly.

Let a set of all APs with maximum gain from s to node $i \in N$ in G_1 be denoted by Q_{si} and its gain by g_{si}. And if there exists no AP from s to i, we set $g_{si} = 0$. The same concern is used for node $j \in N$ to node t. Consider the termination of Proc *Out flow*. Consider $(i,j) \in A_1^c$. If $g_{si} = 0$ (or $g_{jt} = 0$), clearly there will be no s-t AP by adding (i,j). If $g_{si} > 0$ and $g_{jt} > 0$, then $Q_{si} \bigcup \{(i,j)\} \bigcup Q_{jt}$ is the maximum gain AP with (i,j) and note that Q_{si} and Q_{jt} have no arc in common

(otherwise there exists an AP from s to t in G_1, contradicting the maximality of the flow in G_1). When all the created $s-t$ APs are augmented, we look for the cycles created by (i,j) (if (i,j) is not saturated yet). So, Q_{ji} is determined. Arc (i,j) along with Q_{ji} will form cycles. Every cycle C has the same gain factor. If the gain factor $g(C) \leq 1$, arc (i,j) is added to A_1^c, otherwise (i,j) is kept in A_1^c. In the latter case, there exists FGCs in $G_1 \cup (i,j)$. Using Proc *Layer* one can determine paths in Q_{ji} that contains the smallest number of arcs which will define an FGC with the arc (i,j). Then Q_{jt} is determined. Proc *Layer* on Q_{jt} will define paths with the smallest number of arcs. Finally, there is a GAP $(Q_{ji}-i-j-Q_{jt})$.

Note that if $(i,j) \in A_1^c$ forms an FGC with the arcs in G_1, the addition of the arc to G_1 will create problems in determining the maximum gain subnetwork. Then, arc (i,j) is kept in A_1^c until it is saturated or until all the cycles formed by (i,j) are FAC or UGC.

All the above procedures are done by Algorithms 3, 4 and 5 given in Appendix.

3.4 Determine the maximum out flow in $G_1 \bigcup G_1^c$

When Step 4 in Algorithm 1 is finished, apply search algorithm [2] for $(G_1 \bigcup G_1^c)^b$. Then, we can find all the nodes that can reach node t along APs. Let these nodes be shown by N_4. We define subnetwork $G_4 = (N_4, A_4)$ in which $A_4 = \{(i,j), (\overline{i,j}) \in G_1 \bigcup G_1^c : i,j \in N_4\}$.

In Algorithms 6 and 7, we first detect MGAPs in G_4 and saturate (augment) them to increase the excess of t. Next, if any GAPs are created, they are immediately saturated (this is done in Step 9 of Algorithm 7) and then we look for the new MGAPs. These sequences are repeated until there are no MGAPs.

To detect MGAPs, we should detect UGCs and FACs (as the basis of the multiple cycles) in G_4. Wayne [46] has designed a polynomial time algorithm based on Bellman-Ford algorithm to identify bicycles if one exists; if there is none, then UGCs. At the beginning of Algorithm 6, there are no FGCs and consequently G_4 has no bicycles, then by Wayne's algorithm we can obtain a subset, say N', of all nodes that either participate in an FAC or can reach one along a path and afterwards since there are no bicycles we can also find all nodes on UGCs and add them to N'. Let G' denote a subnetwork of G_4 induced by N'. Then all arcs by one outdegree are eliminated from G'. Next, by a depth first search [2] we can detect a cycle (an FAC or a UGC) in G'. The detected FAC and UGC can be the basis of an MGAP.

To detect them polynomially, we use Wayne's algorithm [46]. In fact, his algorithm is designed to detect all bicycles and UGCs in $O(nm)$ time. His basic approach is to first identify a bicycle if one exists; if we don't find one, then we detect UGCs. His algorithm first detects the FACs and then FGCs connected to the detected FACs. Then, if there are no bicycles, it detects UGCs. Here, at first, G_4 has no bicycles (otherwise there is a GAP and it contradicts the condition that we saturate all the GAPs before). But, after the first augmentation some FGCs might be created in G_4 by appeared real arcs. As soon as they are created, if they are part of some GAPs, they are saturated. Otherwise they cannot reach t.

Then, they do not belong to the updated G_4. Consequently, the updated G_4 has also no bicycles and we are able to apply Wayne's algorithm during the execution of Algorithm 6.

We design Proc*Blocking Arcs* given in Appendix to know the amount of augmentation of a path or cycle and their blocking arcs.

Algorithm 6: Determining the maximum outflow in $G_1 \bigcup G_1^c$
Input: $G_1 \bigcup G_1^c$.
Output: maximum outflow in $G_1 \bigcup G_1^c$.

1. Apply search algorithm for $G_1 \bigcup G_1^c$ to make G_4.

   (1.1) **If** $N_4 \neq \oslash$ and G_4 contains some parallel and unmarked arcs, **then** go to Step 2; **else** stop. The current flow is the generalized maximum outflow.

2. Use Wayne's algorithm [46] in G_4 to detect all nodes, N', participated in FACs and UGCs. Make subnetwork G' of G_4 induced by N'.

   (2.1) Use depth first search in G' to detect a cycle.

   (2.2) **If** G' has no cycles or has no cycle with some unmarked arcs (i.e. $\mathbf{G^r}$ has no MGAP), **then** stop. The current flow is the generalized maximum outflow.

   (2.3) **If** the cycle found is simple, mark its arcs and go to Step 2.1.

3. Use Proc *Multiple Cycles*. **If** the procedure returns $\oslash$, **then** go to Step 2.1; **else** go to Step 1.

Algorithm 7: Multiple Cycles
Input: a multiple cycle named C.
Output: $\oslash$ if C is not part of an MGAP. Otherwise the detected MGAP and created GAPs are saturated

1. $\overline{C} = C$. Use Proc *Blocking Arcs* for the basis of C named C_1 to distinguish its blocking arcs and its amount of augmentation shown by BA_1 and α_{C_1}, respectively.

   (1.1) **If** BA_1 is the subset of the parallel arcs of C, **then** go to Step 2; **else** mark arcs of C and return $\oslash$.

2. **If** $t \in N(C_1)$ (nodes of C_1), **then** set $\alpha_P = \infty$ and go to Step 4; **else** let i_C be the junction node and let $P = Q_{i_C t}$ ($Q_{i_C t}$ is found by Bellman-Ford algorithm.

3. Use Proc *Blocking Arcs* for P to calculate its amount of augmentation, α_P.

4. (**If** $\alpha_{C_1} = 0$, **then** use Proc *Blocking Arcs*) $\gamma_1 = \alpha_{C_1}(g(C_1) - 1)$ and $A = A_{C_1} \bigcup A_P$ (arcs of C_1 and P).

5. **For** $l = 1$ to $|K|$ **do** (K is the set of parallel arcs of C)

   (5.1) $d = l$ and $g(C_{l+1}) = \frac{g(C_l)}{g(BA_l)^2}$.

   (5.2) $A_C = A_C - BA_l$. Consider C_{l+1} as new basis.

   (5.3) Use Proc *Blocking Arcs* for C_{l+1} to know the BAs of C_{l+1} and its augmentation α_{l+1}.

   (5.4) $\gamma_{l+1} = \alpha_{C_{l+1}}(g(C_{l+1}) - 1) + \gamma_l$.

   (5.5) **If** $\gamma_{l+1} < \alpha_P$, **then** $l = l + 1$. Let K_l denote the parallel arcs of C (K_l could be different from K).
   If $BA_l \subseteq K_l$, **then** go to the start of Step 5;
   else $d = l$ and go to Step 6;
   else $A = A \bigcup_{1 \leqslant j \leqslant l+1} A_j$, $d = l + 1$ and go to Step 8.

6. $m = Max_{1 \leqslant i \leqslant l}\{\gamma_i\}$.

7. **If** $m > 0$, **then** let r be the minimum index that $\gamma_r = m$. $A = A \bigcup_{1 \leqslant j \leqslant r} A_j$ is an MGAP. Go to Step 8; **else** mark arcs of $\overline{C}$ and return $\oslash$.

8. Use Proc *Advanced block flow* to determine the blocking flow in A.

   (8.1) Use Proc *Update Cap* by blocking flow to update arc capacities. F=F+ BF.

9. **While** $K^R \neq \oslash$ **do** (K^R is the set of appeared real arcs of K)

   (9.1) Select $(i, j) \in K^R$. Find Q_{ji} using Bellman-Ford algorithm.

   (9.2) Find the layered network using Proc *Layer*.

   (9.3) **If** $g(Q_{ji})\gamma_{ij} \leq 1$, **then** remove (i, j) from K^R and go to Step 9; **else** determine Q_{jt} using Bellman-Ford algorithm.

   (9.4) Let j_c denote the junction node between the cycle $Q_{ji} \bigcup (j, i)$ and Q_{jt}.

   (9.5) Find the layered network using Proc *Layer* with the junction node j_c replacing node s.

   (9.6) Use Proc *Advanced block flow* for the created FGC and for the multiple AP .

   (9.7) Use Proc *Cap alt.* Remove (i, j) from K^R.

10. Stop.

Proposition 4. *After the capacity update routine (Proc Update Cap), the reverse arcs will not create a GAP.*

Proof. There are two cases:

First, for Q_{it} (i is a junction node on an MGAP), the reverse arcs after the capacity update will not create any FGCs. This is proved similar to proposition 1 since Q_{it} has the maximum gain factor along all the available $i-t$ APs.

Second, consider C as a multiple cycle (as a part of an MGAP). In Algorithm 7, we carry on augmenting flow through some subcycles of C until the rest of the subcycles will definitely not create any excess or, the nodes of C have no connection (AP) to t.
In the former, let L be the last saturated subcycle of C. Undoubtedly, L is an FGC and its subsequent subcycle named S is a UGC or FAC (otherwise L is not the last subcycle. If C is completely saturated, S is empty). As a result, the reverse arcs of L as a basis of a new multiple cycle or as a simple cycle will be an FAC and arcs of S as a probable basis of a new multiple cycle or as a simple cycle are not an FGC.
The latter, even if S is an FGC, since there are no APs from C to t, there is no GAP.

□

The following proposition proves the finiteness of the algorithm.

Proposition 5. *No MGAP is used more than once .*

Proof. Imagine Γ is an MGAP with a multiple cycle C and an AP from C to t. Let C_k denote k_{th} subcylce of C, $1 \leq k \leq l$. If C_1, ..., C_k ($k \leq l$) are saturated, the reverse of all C_k, ..., C_1 will not be selected in Algorithm 7 since they together create deficits. Consequently, Γ is not created again. □

According to Theorem 2, the proposed algorithm correctly solves the maximum outflow in **G**.

4 Procedure advanced block flow

In this paper, in order to determine the blocking flow we propose and use a procedure named Advanced block flow which is entirely different from Pulat's procedure named Block flow in [36]. In fact, the difference occurs because of the difference in augmenting flow through the multiple and simple residual APs and cycles. The difference is re-described by the following example.

Figure 3 shows a multiple AP with basis $\overline{s-1-2}-\overline{3-t}$ named P_1. Then, depending on which arc will be saturated after augmentation, a new path but with the same sets of nodes as P_1 might exist. Suppose for instance $(\overline{s,1})$ is the blocking arc first. Then, new path P_2 (as a subpath), $s-\overline{1-2}-\overline{3-t}$, exists. During the MPA, if necessary, P_2 might be detected and augmented to create extra excess at t. Consequently, some procedures such as Bellman-Ford algorithm, proc *Layer*, *Advanced block flow*, and *Cap alt* are repeated several times to saturate all the $s-t$ APs as the subpaths of a multiple AP such as P_1 and P_2. To avoid repeating

the procedures, we propose Proc *Advanced block flow* (see an example 4.2). So, the purpose of Proc *Advanced block flow* is to augment flow through the multiple APs and MGAPs instead of augmenting flow separately in their basis and the subsequent subpaths and subcycles.

Note that if there is an FGC and it is a basis of a multiple residual cycle, then we use Proc *Advanced block flow* just for the FGC. Otherwise, some deficit might create or the amount of excess created by the FGC might be decreased (Proc *Advanced block flow* is also applicable for simple APs).

4.1 The correctness

We just describe the reasons for APs and we have the same concern for the cycles of MGAPs.

Undoubtedly, utilizing Proc *Advanced block flow* only once seems more beneficial than applying Proc *Block flow* beside several repetitions of procedures such as Bellman-Ford algorithm, *Layer* and *Cap alt*. The running time of all procedures required to augment flow through a simple AP either by Proc *Block flow* or Proc *Advanced block flow* is $O(n^2m)$. To augment flow through a multiple AP by Proc *Block flow*, maximum $m+1$ iterations of Bellman-Ford algorithm, *Layer* and *Cap alt* are needed. However, the iterations reduce to one by applying Proc *Advanced block flow*. Consequently, the complexity of the two mentioned cases are $O((m+1)n^2m)$ and $O(n^2m)$, respectively. As a result, it is estimated that if a multiple AP contains lots of parallel arcs (then the multiple AP contains many subpaths) or is too long, the proposed algorithm including Proc *Advanced block flow* is more efficient.

Now we should prove that after applying the Proc *Advanced block flow* and Proc*Cap alt*, no FGCs will be created. So we need Proposition 7. It is given and proved in Section 3.2.

Consider α_i denotes the maximum amount of flow that node i can send to its subsequent node to augment flow in an AP. Furthermore, let $\overline{u}_{ij}$ show the capacity of $(\overline{i,j})$ ($\overline{u}_{ij} = f_{ji}\gamma_{ji}$). In addition, two functions named Heaviside step and Sign are used in the following procedure and shown by H and Sgn respectively. The Proc *Advanced block flow* is designed for $s-t$ APs. Then, for any $i-j$ APs (or cycles), i and j are replaced by s and t, respectively.

Algorithm 8: Advanced block flow
Input: $s-t$ APs or residual cycles .
Output: the blocking flow.

1. Initialize: set k = 1 and w(k) = s, where w(k) denotes the k-th node in the AP (multiple or simple), henceforth referred to as an element of path. All arcs of the input network, A_3, are 'unmarked' initially. Go to Step 2 with $i = s$.

2. **If** there exists an unmarked arc (i,j) or $(\overline{i,j}) \in A_3$, **then**

(2.1) $k = k + 1$, $w(k) = j$, $i = j$.

(2.2) **If** $i \neq t$, **then** go to the start of Step 2; **else** the AP found is named P_s and go to Step 3;

else backtrack to node $w(k-1)$.

(2.3) **if** $k = 0$, **then** go to Step 7; **else** let $h = w(k-1)$, mark arc (h, i), set $i = h$, $k = k - 1$ and go to Step 2.

3. /* Construct the multiple AP named P_m*/
P_m includes P_s and $(i, j) \in A_1$ where $(\overline{i, j}) \in P_s$. Go to Step 4 by P_m.

4. /* Augment flow into node t*/

(4.1) $\alpha_{w(1)} = \overline{u}_{w(1)w(2)} + u_{w(1)w(2)}, l = 2$.

(4.2) **While** $l \leq k$ **do**

i. $i = w(l-1), j = w(l)$. **If** $l \neq k$, **then** $h = w(l+1)$.

ii. $\beta_j = (\frac{\overline{u}_{ij}}{\gamma_{ij}} + (\alpha_i - \overline{u}_{ij})\gamma_{ij} Sgn(\alpha_i - \overline{u}_{ij}))H(\alpha_i - \overline{u}_{ij}) + \frac{\alpha_i}{\gamma_{ij}} Sgn(\overline{u}_{ij} - \alpha_i)H(\overline{u}_{ij} - \alpha_i)$.

iii. **If** $l = k$, **then** $\alpha_j = \beta_j$; **else** $\alpha_j = min(\beta_j, \overline{u}_{jh} + u_{jh})$.

iv. $l = l + 1$.

5. /* Modify capacity*/ $\gamma = 1$.

(5.1) **While** $k \geq 2$ **do**

i. $i = w(k-1), j = w(k), \theta = \alpha_j$.

ii. **If** $u_{ij} > 0$ and $\overline{u}_{ij} > 0$, **then** $R = u_{ij}$ and $r = 1$ **else** $r = 0$.

iii. $\overline{u}_{ji} = \overline{u}_{ji} + (\theta\gamma_{ij} - \overline{u}_{ij})\gamma_{ij} Sgn(\gamma_{ij}\theta - \overline{u}_{ij})H(\gamma_{ij}\theta - \overline{u}_{ij})$.

iv. $u_{ji} = u_{ji} + \theta H(\overline{u}_{ij} - \gamma_{ij}\theta)$.

v. **If** $\overline{u}_{ij} = 0$, **then** $u_{ij} = u_{ij} - \frac{\theta}{\gamma_{ij}}$;
else $u_{ij} = u_{ij} - (\theta\gamma_{ij} - \overline{u}_{ij})Sgn(\gamma_{ij}\theta - \overline{u}_{ij})H(\gamma_{ij}\theta - \overline{u}_{ij})$.

vi. $\overline{u}_{ij} = (\overline{u}_{ij} - \gamma_{ij}\theta)H(\overline{u}_{ij} - \gamma_{ij}\theta)$.

vii. **If** $R > u_{ij}$ and $r = 1$, **then** $\gamma = \gamma\gamma_{ij}$.

viii. $k = k - 1$.

6. **If** $\gamma < 1$, **then** use original capacity vector, U, and P_s. Go to Step 4; **else**

(6.1) mark each arc (i, j) or $(\overline{i, j})$ whose capacity is zero.

(6.2) **If** there are unmarked arcs, **then** let $w(k')$ denote the tail of the last unmarked arc on the path. Set $h = w(k')$, $i = h$, $k = k'$ and clear j. Delete arcs in P_s from k' to t. Then return to the start of Step 2; **else** go to Step 7.

7. /* Determine blocking flow in G_3*/ Let U and U_2 denote the original capacity vector and the capacity vector at this step, respectively. The blocking flow vector in G_3 is determined from blocking flow $= U - U_2$.

4.2 An example

Figure 4 is provided to perform a comparison between Proc *Advanced block flow* and Proc*Block flow*.

In Figure 4, four sequential APs are $\overline{1-2-3-4}$ (the basis), $1-\overline{2-3-4}$, $1-2-\overline{3-4}$ and $1-2-3-4$ when 2,3,2 and 2 units of flow are augmented through them, respectively.

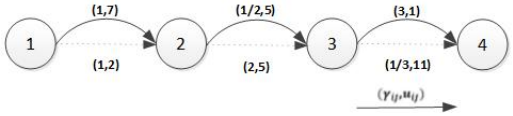

Figure 4: A multiple AP in $\mathbf{G}^r$

Now consider Proc*Block flow* is used in the MPA. After using Bellman-Ford algorithm and Proc *Layer*, if the basis is selected to be augmented, then the next residual AP is $1-\overline{2-3-4}$ shown in Figure 5. Suppose that this path is also selected to be augmented, then the next AP is $1-2-\overline{3-4}$. After saturating it, if it is in G_3, next AP $1-2-3-4$ will be detected. Finally the excess $\frac{20}{3}$ creates at t if the last AP belongs to G_3 too. Consequently, several repetitions of Proc *Layer*, Bellman-Ford algorithm, *Block flow* and Proc *Cap alt* are necessary for all the preceding operations.

Now consider Proc*Advanced block flow* is used in the MPA. Since the multiple AP satisfies equation 4.1 (property of Proposition 7), it is possible to saturate four APs at the same time by augmenting 9 units of flow and achieve $\frac{20}{3}$ excess at t as well. Notice that even if the AP in Figure 5 is not in G_3 , i.e., it is not allowed to be augmented and as a result the next two APs will not exist in $\mathbf{G}^r$ in the preceding paragraph, applying Proc *Advanced block flow* and augmenting flow in the multiple AP are considered more beneficial than applying Proc*Block flow* and augmenting flow just in its basis since the first case creates more excess at t in less time.

Figure 5: The AP in G_3

5 The other example

Take undirected network G in Figure 6 to apply the MPA.

The s-t path found in Proc*Outflow in* G_1 is s-1-3-2-t.

Now arc (2,1) is chosen from G_1^c. In fact, this is the only arc from G_1^c made a GAP. Then $G_1 \bigcup G_1^c$ has an MGAP with the basis in Figure 9. Its subsequent cycles are shown in Figure 10 and 11.

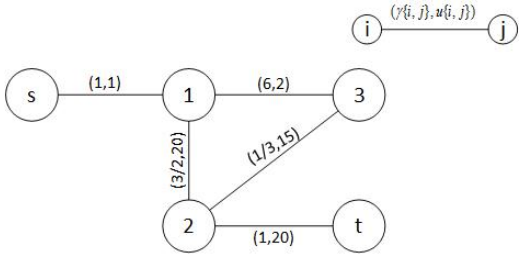

Figure 6: Undirected network G

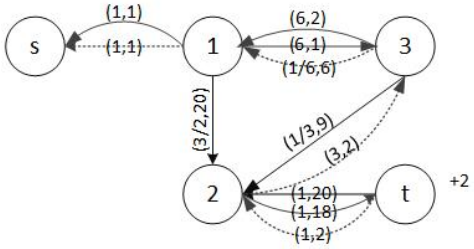

Figure 7: The output of Proc *Outflow in* G_1

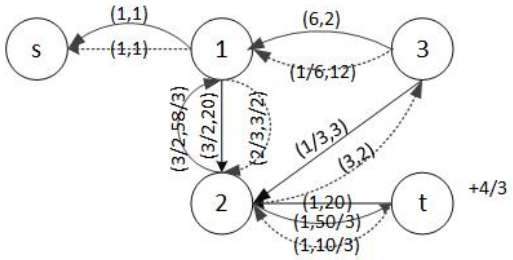

Figure 8: The output of Proc *Out flow in augmented* G_1

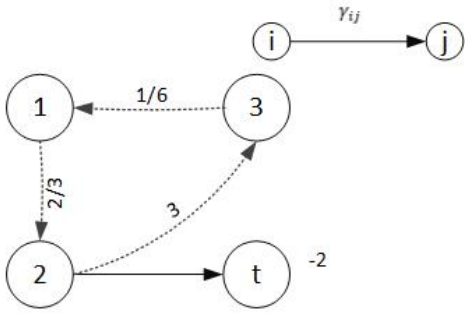

Figure 9: An FAC and an AP to t in $G_1 \bigcup G_1^c$

Finally, $G_1 \bigcup G_1^c$ satisfies the optimality condition in Theorem 2 ($v_t = 2 + 4/3 - 2 - 1/2 + 12 = 77/6$).

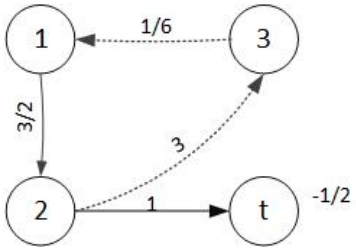

Figure 10: The FAC and the AP to t in $G_1 \bigcup G_1^c$

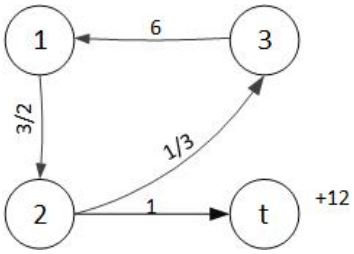

Figure 11: The FGC and the AP to t in $G_1 \bigcup G_1^c$

6 Conclusions

We have designed an algorithm for undirected generalized maximum out flow problem. We have also stated and proved the optimality condition of maximum outflow problem in its corresponding directed network. In addition, we have improved its running time by a complete change in one of its procedure. The proposed algorithm raises several interesting questions for future research such as the maximization of the value of flow from s to t (inflow), the maximization of flow in multi-terminal networks, multi-commodity problems, convex generalized flow,or design problems.

7 Appendix

This Section is about Proc *Out flow in augmented* G_1, Proc *Max path flow in aug* G_1, Proc *Cycle max flow in aug* G_1 and Proc *Cap alt* from Pulat [36] adapted to GMOPUN. We also design Proc *Blocking arcs* at the end.

Algorithm 3: Proc Out flow in augmented G_1*- the parent Proc*
Input: A_1 and A_1^c.
Output: $A_1 \bigcup A_1^c$ without any $s-t$ APs and GAPs.
Initially all arcs are unscanned.

1. **If** A_1^c or all the arcs in A_1^c are scanned, **then** stop;
   else unscan all arcs and go to Step 2.

2. Select $(i,j) \in A_1^c$. Use Proc *Max path flow in aug* G_1.

If $g_{jt} \neq 0$ and $u_{ij} > 0$, **then** go to Step 3;
else scan arc (i, j) and go to Step 1.

3. Use Proc *Cycle max flow in aug* G_1. Go to Step 2.

Algorithm 4: Proc Max path flow in aug G_1
Input: this Proc is entered with $g_{si} > 0$ (corresponding to subnetwork of maximal gain Q_{si}) and $g_{jt} > 0$ (corresponding to subnetwork of maximal gain Q_{jt}).
Output: A_1' without any $s - t$ APs.

1. Go to Proc *Layer* with Q_{si} and Q_{jt}.

2. **If** no layered network can be constructed for either network, **then** return to Parent Proc; **else** go to Proc *Advanced block flow*, then go to Step 3.

3. Update arc capacities in Q_{si}, Q_{jt} and arc (i, j) using Proc *Cap alt.*
If $u_{ij} = 0$, **then** add arc (i, j) to A_1 and subtract it from A_1^c. Return to Parent Proc; **else** determine new $g_{si} > 0$ and $g_{jt} > 0$, and go to Step 1.

Algorithm 5: Proc cycle max flow in aug G_1
Input: A_1 with arc $(i, j) \in A_1^c$.
Output: A_1' without GAPs.

1. With arc $(i, j) \in A_1^c$, determine Q_{ji}, hence g_{ji} using Bellman-Ford algorithm.
If no maximum gain path exists from node j to node i, **then** label arc (i, j) as scanned and return to Parent Proc.
else if $g(C) \leq 1$ $(g(C) = g_{ji}\gamma_{ij})$, **then** set $A_1 = A_1 \cup (i, j)$ and $A_1^c = A_1^c - (i, j)$ and return to start of the step; **else** go to Step 2.

2. Find the layered network using Proc *Layer* with the nodes j and i replacing the nodes s and t, respectively.
If no layered network exists, **then** go to Step 1; **else** go to Step 3 with $C = \{\text{arcs of the layer network} \cup (i, j)\}$.

3. Find the maximum gain subnetwork Q_{jt}, using Bellman-Ford algorithm.
If no such path exists, **then** go to Step 1; **else** go to Step 4.

4. Let j_c denote the junction node between the cycle C and the maximum gain subnetwork Q_{jt}. Find the layered network $G_3 \subset G_2$ using Proc *Layer* with the junction node j_c replacing node s.
If there exists no layered network, **then** go to Step 3; **else** continue to Step 5.

5. Determine the blocking flow using Proc *Advanced block flow.* Maximum flow through the cycle to node j_c (for the basis) and from node j_c to node t is determined.

6. Perform capacity alteration by blocking flow in G_2 and G_1 using Proc *Cap alt*.
   If the saturated arc is in G_3, **then** go to Step 4.
   else if $u_{ij} > 0$, **then** go to Step 1.
   else $u_{ij} = 0$, in which case set $A_1 = A_1 \cup (i, j)$. Return.

Algorithm 9: Proc Cap alt
To alter the capacity of the arcs in the maximum gain subnetwork by the blocking flow in the layered subnetwork.

1. **For** arc (i, j) with $f_{ij} > 0$ (blocking flow), $u_{ij} = u_{ij} - f_{ij}$ and $\overline{u}_{ji} = \overline{u}_{ji} + \gamma_{ij} f_{ij}$.
   For arc (i, j) with $f_{ij} = 0$, u_{ij} and $\overline{u}_{ji}$ remain unchanged.

Algorithm 10: Blocking arcs
Input: Arcs of an AP or a cycle (GAP) that are denoted by A.
Output: The amount of augmentation of the inputs and their blocking arcs.

1. Unscan all arcs of A.
2. i=s, $\beta_i = \infty$, $l = 1$.
3. **If** there is no $(j, i) \in$ A, **then** go to Step 5; **else** go to Step 4.
4. Select an $(i, j) \in A \setminus P$.
   (4.1) **If** j=t, **then** $P = P \bigcup \{(i, j)\}$, set $i = s$ and mark arcs $(s, h) \in P$. Go to the start of Step 4.
   (4.2) **If** j=s, **then** $P = P \bigcup \{(i, j)\}$, select an unmarked arc $(s, h) \in P$, i=s, j=h and unmark all arcs. Start Step 5 by (i, j).
   (4.3) **If** $j \neq s$ and $j \neq t$, **then** $P = P \bigcup \{(i, j)\}$, i=j and clear j. Go to the start of Step 4.
5. Select an unscan $(i, j) \in A$ and continue. **If** there is none, **then** go to Step 6.
   (5.1) **If** i=s and $l > 1$, **then** $\alpha_{ij} = \alpha_s$; **else** $\alpha_{ij} = min\{\beta_i, u_{ij}\}$.
   (5.2) **If** $\alpha_{ij} = u_{ij}$ and $i \neq s$, **then** $K = K \bigcup \{\alpha_{ij}\}$.
   (5.3) $\beta_j = \alpha_{ij} \gamma_{ij}$ and $L = L \bigcup \{(i, j)\}$.
   (5.4) Scan arc (i, j).
   (5.5) **If** j=t, **then**
   If $\alpha_s = 0$, **then** for $(s, h) \in L$ set $\alpha_{sh} = \frac{\beta_j}{\prod_{(v,w)\in L} \gamma_{vw}}$.
   (5.6) **If** $\alpha_{sh} = u_{sh}$, **then** $K = K \bigcup \{\alpha_{sh}\}$, $\alpha_s = \alpha_{sh}$ and go to Step 6.
   (5.7) **else** for the marked $(s, h) \in L$ set $\alpha_{sh} = \frac{\beta_j}{\prod_{(v,w)\in L} \gamma_{vw}}$.

(5.8) **If** $\alpha_{sh} = u_{sh}$, **then** $K = K \bigcup \{\alpha_{sh}\}$, $\alpha_s = \alpha_{sh}$. Unmark (s, h) and go to Step 6.

(5.9) **If** j=s, **then** for $(s, h) \in L$ set $\alpha_{sh} = \frac{\beta_j}{\prod_{(v,w)\in L} \gamma_{vw}}$, mark (s, h), $\alpha_s = \beta_s - \alpha_{sh}$, $l = l + 1$ and go to the start of Step 5.

(5.7) $l = l + 1$ and go to the start of Step 5.

6. $\alpha = \min\{\alpha_{ij} : (i, j) \in K\}$.
for all $(i, j) \in L$ **if** $\alpha_{ij} = \alpha$, **then** mark (i, j) as the blocking arc.
Return α_s as the amount of augmentation and marked arcs of L as the blocking arcs.

References

[1] Adams, Beglari, Laughton, and Mitra, *Mathematical programming systems in electrical power generation, transmission and distribution planning*, Proc. with Power System Computation Conference, 1972.

[2] Ahuja, R. K., Magnanti, T. L. and Orlin, J. B., *Network Flows, Theory, Algorithms, and Applications*, Prentice Hall, Englewood Clifs, NJ, 1993.

[3] Balas, E., and Evanescu, P.L., *On the generalized transportation problem*, Management Science **11** (1964), 188-202.

[4] Balas, E., *The dual method for the generalized transportation problem*, Management Science **12** (1966), 555-568.

[5] Barzilia, J., Cook, W.D., and KRESS, M., *A generalized network formulation of the pairwise comparison consensus ranking model*, Management Science **32** (1986), 1007-1014.

[6] Bhaumik, G., *Optimum operating policies for a water distribution system with losses*, Unpublished PhD dissertation, University of Texas at Austin, Texas, (1973).

[7] Busacker, R. G. and Gowen, P. J., *A procedure for determining a family of minimum-cost network flow patterns*, Technical Report 15, Operations Research Office, Johns Hopkins University, (1961).

[8] Charnes, A., and Cooper, W.W., *Management Models and Industrial Applications of Linear Programming*, Wiley, New York, (1961).

[9] Crum, R.L., Klingrnan, D., and Tavis, L.A., *Implementation of large-scale financial planning models; Solution efficient transformations*, Research report CCS 267, The University of Texas at Austin, Texas, (1976).

[10] Crum, R.L. and Nye, DJ., *A network model of insurance company cash flow management*, Mathematical Programming **15** (1981), 86-101.

[11] Dror, M., Trudeau, P. and Ladany, S.P., *Network models for seat allocation on flights*, Transportation Research **22B** (1988), 239-250.

[12] Eiseman, K., *The generalized stepping stone method for the machine loading model*, Management Science **11** (1963), 154-176.

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

[14] Fulkerson, D. R., *An out-of-kilter method for minimal cost flow problems*, SIAM Journal **9** (1961), 13-27.

[15] Glover, F., Glover, R. and Martinson, F.K., *A netform system for resource planning in the U.S. Bureau of land management*, Journal of the Operational Research Society **35** (1984), 605-616.

[16] Glover, F., Hultz, J., Klingman, D., and Stutz, J., *Generalized networks: A fundamental computer based planning tool*, Management Science **24/12** (1978), 1209-1220.

[17] Glover, F., Klingman, D., and Phillips, N., *A new polynomially bounded shortest path algorithm*, Operations Research **33** (1985), 65-73.

[18] Glover, F., Klingman, D., and Phillips, N., *Netform modelling and applications*, Interfaces **20** (1990), 7-27.

[19] Glover, F., and Rogozinski, J., *Resort development: A network-related model for optimizing sites and visits*, Journal of Leisure Research (1982), 235-247.

[20] Goldberg, A. V., Plotkin, S. A. and Tardos, E'., *Combinatorial algorithms for the generalized circulation problem*, Mathematics of Operations Research **16** (1991), 351- 379.

[21] Goldfarb, D. and Jin, Z., *A faster combinatorial algorithm for the generalized circulation problem*, Mathematics of Operations Research **21** (1996), 529-539.

[22] Gondran, M. and Minoux, M., *Graphs And Algorithms*, Wiley, 1984.

[23] Gorham, W., *An application of a network flow model to personnel planning*, EEE Transactions on Engineering Management **10** (1963), 113-123.

[24] Guim, L., and Nye, D.J. , *A network model of insurance company cash flow management*, Mathematical Programming Study **15** (1981), 86-101.

[25] Horowitz, E., and Sahni, S., *Fundamental of Computer Algorithms*, Computer Science Press, Inc, (1974).

[26] Iri, M., *A new method of solving transportation-network problems*, Journal of the Operations Research Society of Japan **3** (1960), 27-87.

[27] Itai, A., *Two-commodity flow*, Journal of the Association for Computing Machinery **25** (1978), no. 4, 596-611.

[28] Jarvis, J. J. and Jezior, A. M., *Maximal flow with gains through a special network*, Operations Research. **20** (1972), 678-688.

[29] Jewell, W. S., *Optimal flow through networks*, Technical Report 8, Operations Research Center (1958) MIT.

[30] Jewell, W. S., *Optimal flow through networks with gains*, Operations Research **10** (1962), 476-499.

[31] Kim, Y., *An optimal computational approach to the analysis of a generalized network of copper refueling process*, Joint ORSA /TIMS/AIIE Conference, Atlantic City, NJ, (1972).

[32] Klein, M., *A primal method for minimal cost flows with applications to the assignment and and transportation problems*, Management Science **14** (1976), 205- 220.

[33] Liu, C., and Wu, F.F., *A generalized network flow model with application to power supply-demand problems*, Networks **14** (1984), 117-139.

[34] Olver, N. and Vgh, L. A., *A simpler and faster strongly polynomial algorithm for generalized flow maximization*, In Proceedings of 49th Annual ACM SIGACT Symposium on the Theory of Computing, Montreal, Canada, (STOC17) (2017), no. 12.

[35] Onaga, K., *Dynamic programming of optimum flows in lossy communication nets*, IEEE Trans, Circuit Theory **13** (1966), 308-327.

[36] Pulat, P.S., *Maximum outflow in generalized flow networks*, European Journal of Operational Research **43** (1989), 65-77.

[37] Radzik, T., *Faster algorithms for the generalized network flow problem*, Mathematics of Operations Research **23** (1998), 69-100.

[38] Restrepo, M. and Williamson, D.P., *A simple gap-cancelling algorithm for the generalized maximum flow problem*, Mathematical Programming **118** (2009), 47-74.

[39] Robichek, A.A., Teichroew, D., and Jones, J.M., *Optimal short-term financing decision*, Management Science **12** (1965), 1-36.

[40] Sasson, A.M. , *Nonlinear programming solutions for load flow, minimum loss and economic dispatching problems*, IEEE Transactions on Power Apparatus Systems PAS. **88** (1969), 399-409.

[41] Steinberg, E., and Napier, H.A., *Optimal multi-level lot sizing for requirements planning systems*, Management Science. **26/12** (1980), 1258-1271.

[42] Talukdar, S.N., and Morton, T.E., *The optimal operation of power distribution networks with dispersed storage, dispersed generation and curtailable loads*, Unpublished paper, (1980).

[43] Truemper, K., *On max flows with gains and pure min-cost flows*, SIAM Journal on Applied Mathematics **32** (1977), 450-456.

[44] Truemper, K., *Optimal flows in networks with positive gains*, PhD thesis, Case Western Reserve University, (1973).

[45] Vegh, L. A, *A Strongly Polynomial Algorithm for Generalized Flow Maximization*, Mathematics of Operations Research (2016), 1-33.

[46] Wayne, K. D., *A polynomial-time combinatorial algorithm for generalized minimum cost flow*, STOC '99 Proceedings of the thirty-first annual ACM symposium on Theory of computing (1999), 11-18.

[47] Wayne, K. D., *Generalized maximum flow algorithms*, PhD thesis, Cornell University, (1999).