LABELING ALGORITHM FOR GENERALIZED MAXIMUM FLOW NETWORK

A. ANTO KINSLEY*,
Associate Professor, Department of Mathematics,
St. Xavier's college (Autonomous), Palayamkottai- 627 002.
Tirunelveli, India.
Phone: 9443446496, Email: antokinsley@ yahoo.com

B. UMA MAHESWARI,
Research scholar, Department of Mathematics,
St. Xavier's college (Autonomous), Palayamkottai- 627 002.
Tirunelveli, India.
Phone: 9442624315, Email: umaraju211199@gmail.com

Abstract— **In any traditional network, there is an implicit assumption that flow is conserved on every arc. In generalized networks, each arc has a positive multiplier $\gamma(u, v)$ called a gain factor, associated with it, representing the fraction of flow that remains when it is sent along that arc. The generalized maximum flow problem is identical to the traditional maximum flow problem, except that it can also model network with "leak" flow. In this paper, we describe the application of the labeling algorithm to the maximum-flow generalized network problem. This algorithm has complexity O (EV), where E and V are the number of edges and vertices respectively.**

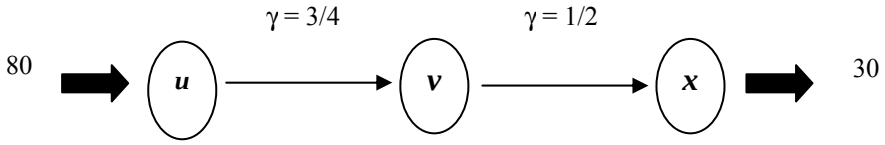
Keywords-Networks; flow; capacity; gain function; maximum flow and generalized networks.

I. INTRODUCTION

For graph theoretic terminology, we refer to Bandy and Murthy [1]. The network flow problem [2] can be solved by several methods. The first algorithm for the network flow problem was given by Ford and Fulkerson [3]. Labeling methods provide an alternative approach for solving network problems. The basic idea behind the labeling procedure is to systematically attach labels to the nodes of a network until the optimum solution is reached.

The generalized maximum flow problem is a natural generalization of the traditional maximum flow problem. In traditional networks, there is an implicit assumption that flow is conserved on every arc. This assumption may be violated if water leak as it is pumped through a pipeline. Many applications are described in [4].

In a generalized network, a fixed percentage of the flow is lost when it is sent along an arc. Specially, each arc (u, v) has an associated gain factor $\gamma(u, v)$. When $f(u, v)$ units of flow enter into the arc (u, v) through node u then $\gamma(u, v)\, f(u, v)$arrive at v. As the example in below illustrates, if 80 units of flow are sent into an arc (u, v) with gain factor 3/4, then 60 units reach node v: if these 60 units are then sent through an arc (v, x) with gain factor ½, then 30 units arrive at x.



The Maximal-Flow Problem

If v denotes the amount of material sent from node s, called the source, to note t, called the sink, the problem can be formulated as follows:

Maximize *v*,

subject to:
$$\sum_j X_{ij} - \sum_k X_{ki} = \begin{cases} v & if\ i = s, \\ -v & if\ i = t \\ 0 & otherwise, \end{cases}$$
$$0 \le X_{ij} \le u_{ij}, \qquad (i = 1, 2, ..., n; \quad j = 1, 2, ..., n).$$

We assume that there is no arc from *t* to *s*. Also, $u_{ij} = +\infty$ if arc $i - j$ has unlimited capacity. The interpretation is that *v* units are supplied at *s* and consumed at *t*.

Letting x_{ts} denote the variable *v* and rearranging, we see that the problem assumes the following special form of the general network problem:

Maximize x_{ts}'

Subject to:
$$\sum_j X_{ij} - \sum_k X_{ki} = 0 \quad (i = 1, 2, ..., n)$$
$$0 \le X_{ij} \le u_{ij}, \qquad (i = 1, 2, ..., n; \quad j = 1, 2, ..., n).$$

Here arc $t - s$ has been introduced into the network with u_{ts} defined to be $+\infty$, x_{ts} simply returns the *v* units from node *t* back to node *s*, so that there is no formal external supply of material.

II. GENERALIZED MAXIMUM FLOW PROBLEM

Definition 2.1

The generalized maximum flow problem is a *generalized network* $G = (V, E, t, c, \gamma, e)$ where *V* is an *n*-set of nodes, *E* is an *m*-set of directed arc, $t \in v$ is a distinguished node called the sink, $c : E \to R \ge 0$ is a capacity function, $\gamma : E \to R \ge 0$ is a *gain function*, and $e : V \to R \ge 0$ is an *initial excess function*. A *residual arc* is an arc with positive capacity. A *flow generating cycle* is a cycle whose gain is more than one.

Definition 2.2

A *generalized pseudo flow* is a function $f : E \to R$ that satisfies the *capacity constraints* $f(v, w) \le C(v, w)$ for all $(v, w) \in E$ and the *anti-symmetry constraints* $f(v, w) = -\gamma(w, v) f(w, v)$ for all $(v, w) \in E$. The *residual excess* of *f* at node *v* is $e_f(v) = e(v) - \sum_{(v,w) \in E} f(v, w)$ i.e., the initial excess minus the net flow out of *v*. If $e_f(v)$ is positive (negative) we say that f has residual excess (deficit) at node *v*. A *generalized flow* is a generalized pseudo flow that has no residual deficits, but it is allowed to have residual excesses. A *proper generalized flow* is a flow which does not generate any additional residual excess, except possibly at the sink.

We note that a flow can be converted into a proper flow, by removing flow on useless paths and cycles. Let *OPT* (*G*) denote the maximum possible value of any flow in network *G*. A flow *f* is *optima*l in network *G* if $|f| = OPT(G)$ and ξ *- optimal* if $|g| \ge (1 - \xi)\ OPT(G)$. The (*approximate*) *generalized maximum flow problem* is to find a (ξ-) optimal flow.

Optimality conditions

An *augmenting path* is a residual path from a node with residual excess to the sink. A *generalized augmenting path* (GAP) is a residual flow-generating cycle, together with a (possibly trivial) residual path from a node on this cycle to the sink. By sending flow along augmenting paths or GAPs we increase the net flow into the sink.

Theorem 2.3

A flow *f* is optimal in network *G* if and only if there are no augmenting paths or GAPs in G_f.

Proof: Refer [3].

III. LABELING ALGORITHM – FORMAL STATEMENT

Initialization

Assume a given feasible flow plan f_{ij} (if none is given, use the feasible plan with all $f_{ij} = 0$). The initial effective capacity u_{ij}^{*} on arc *i*– *j* is given by calculating $c_{ij}^{*} = c_{ij} - f_{ij} + f_{ji}$ (i.e., unused capacity $c_{ij} - f_{ij}$ Plus return capacity f_{ji}).

Path Search

Start with the source node *s* and label (mark) every node *k* with $c_{sk}^{*} > 0$.. Then, in turn, select any labeled node *i* not already scanned (i.e., used to label other nodes) and label every node *j* with $c_{ij}^{*} > 0$ until either *t* has been labeled or no further labeling can take place.

Capacity Update

If *t* has been labeled, then a flow-carrying path *P* has been found from source to sink ($c_{ij}^{*} > 0$ for every arc $i - j$ on the path), and $\theta = Min\{c_{ij}^{*} / i - j \text{ in } P\}$ is the flow capacity of the path. For every arc $i - j$ of *P*, change c_{ij}^{*} to $c_{ij}^{*} + \theta$. i.e., increase the effective capacity of the return path. (Adding or subtracting finite θ to any $c_{ij}^{*} = +\infty$ keeps the c_{ij}^{*} at $+\infty$. If every c_{ij}^{*} in *P* is + ∞, then θ = +∞ and the optimal flow is infinite.)

Termination

If the path search ends without labeling *t*, then terminate. The optimal flow pattern is given by:

$$f_{ij} = \begin{cases} c_{ij} - c_{ij}^{*} & if \quad c_{ij} > c_{ij}^{*} \\ 0 & if \quad c_{ij} \le c_{ij}^{*} \end{cases}$$

Let us consider the water-pipeline system. The numbers above the arcs indicate flow capacity and the bold-faced numbers below the arcs specify a tentative flow plan.

$$X_{ts} = v$$

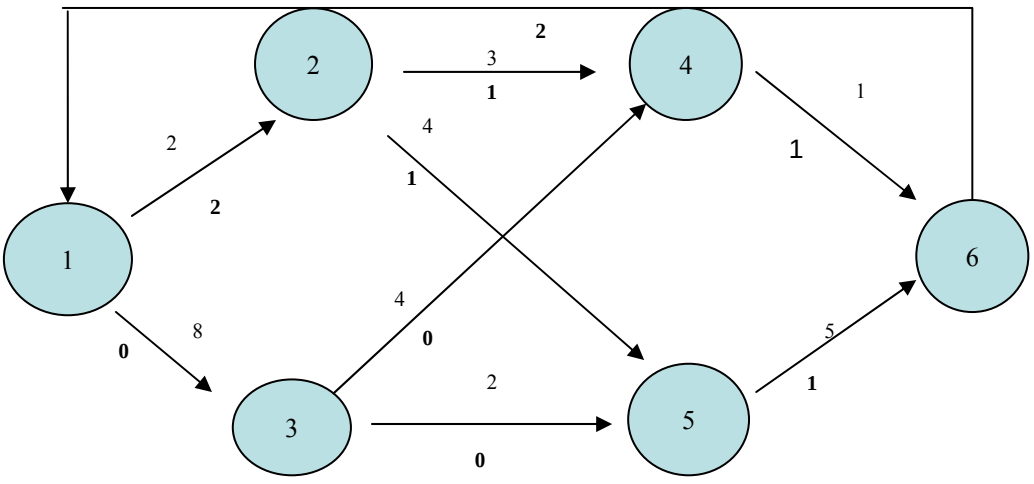
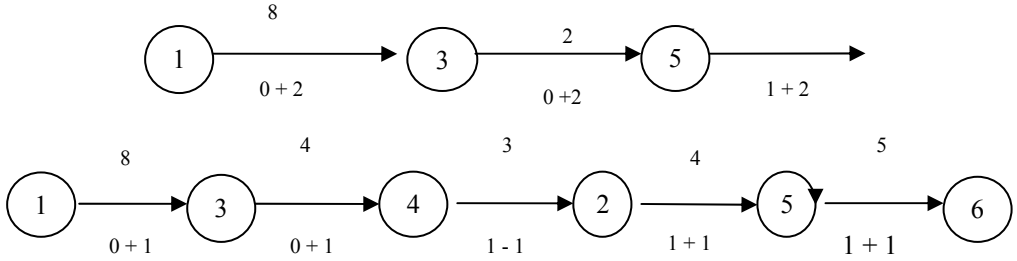


Figure1. A Simple network

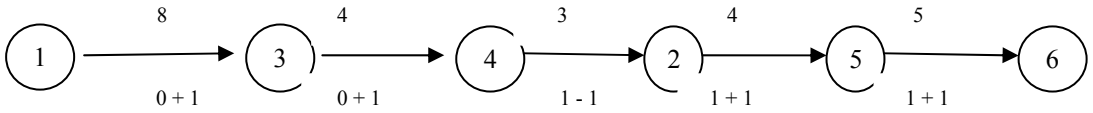
The algorithm for finding maximal flow rests on observing two ways to improve the flow in this example. The following two ‘‘paths’’ appear in Figure1.



In the first case, the directed path 1–3–6 has the capacity to carry 2 additional units from the source to the sink, as given by the capacity of its weakest link, arc 3–5. Note that adding this flow gives a feasible flow pattern, since 2 units are added as input as well as output to both of nodes 3 and 5.

The second case is not a directed path from the source to the sink since arc 2–4 appears with the wrong orientation. Note, however, that adding one unit of flow to the ‘‘forward arcs’’ from 1 to 6 and subtracting one unit from the ‘‘reverse arc’’ 2-4 provides a feasible solution, with increased source-to-sink flow. Mass balance is maintained at node 4, since the one more unit sent from node 3 cancels with the one less unit sent from node 2. Similarly, at node 2 the one additional unit sent to node 5 cancels with the one less unit sent to node 4.

The second case is conceptually equivalent to the first if we view decreasing the flow along arc 2-4 as sending flow from node 4 back to node 2 along the reverse arc 4-2. That is, the unit of flow from 2 to 4 increases the effective *capacity* of the ‘‘return’’ arc 4-2 from 0 in the original network to 1. At the Same time, it decreases the usable or effective capacity along arc 2-4 from 3 to 2. With this view, the second case becomes:



Now both instances have determined a directed flow-*carrying path* from source to sink, that is, a directed path with the capacity to carry additional flow.

The maximal-flow algorithm inserts return, arcs, such as 4–2 here, and searches for flow-carrying paths. It utilizes a procedure common to network algorithms by ‘‘fanning our’’ from the source node, constructing flow-carrying paths to other nodes, until the sink node is reached.

Figure 2 solves the water-pipeline example in Fig. 1 by this algorithm. Checks next to the rows indicate that the node corresponding to that row has already been scanned. The first three tableaus specify the first application of the path – search step in detail. In Tableau 1, node 1 has been used to label node 3. In Tableau 2, node 3 was used to label nodes 4 and 5 (since $C_{34}^{*} > 0$, $C_{35}^{*} > 0$). At this point either node 4 or 5 can be used next for labeling. The choice is arbitrary, and in Tableau 3, node 5 has been used to label node 6. Since 6 is the sink, flow is updated. The last column in the tableau keeps track of how nodes have been labeled. By Backtracking, we get a flow-carrying path *P* from source to terminal. For instance, from Tableau 3, we know that 6 have been labeled from 5, 5 form 3, and 3 from 1, so that the path is 1–3–5–6.

Initial capacity C_{ij}

		1	2	3	4	5	6
Source	1		2	8			
	2				3	4	
	3				4	2	
	4						1
	5						5
Destination	6						

Figure 2. Table for the maximal-flow algorithm

Tableau 1

		Initial capacity C_{ij}						Labeled
		1	2	3	4	5	6	from
Source	1			8				Start
	2	2			2	3		
	3				4	2		1
	4		1					
	5		1				4	
Destination	6				1	1		

Tableau 2

								Labeled
		1	2	3	4	5	6	from
Source	1			8				
	2	2			2	3		
	3				4	2		1
	4		1					3
	5		1				4	3
Destination	6				1	1		

Tableau 3

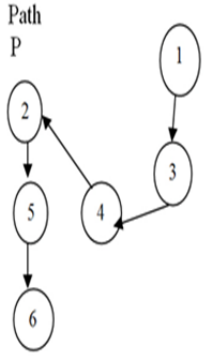
								Labeled	Path
		1	2	3	4	5	6	from	P
Source	1			8					1
	2	2			2	3			
	3				4	2		1	3
	4		1					3	
	5		1				4	3	5
Destination	6				1	1		5	6

$\theta = \underset{i-j \text{ in } P}{Min}\{C_{ij}^{*}\} = Min\{8,2,4\} = 2$. Subtract 2 from C_{13}^{*}, C_{35}^{*}, C_{56}^{*} and Add 2 to C_{31}^{*}, C_{53}^{*} and C_{65}^{*}.

Path 1 – 3 – 5 - 6

Tableau 4

		1	2	3	4	5	6	Labeled from
Source	1			6				start
	2	2			2	3		4
	3	2			4			1
	4		1					3
	5		1	2			2	2
Destination	6				1	3		5



$\theta = \underset{i-j\ in\ P}{Min}\{C_{ij}^{*}\} = Min\{6,4,1,3,2\} = 1$. Subtract 1 from C_{13}^{*}, C_{34}^{*}, C_{42}^{*}, C_{25}^{*}, C_{56}^{*} and Add 1 to C_{31}^{*}, C_{43}^{*}, C_{24}^{*}, C_{52}^{*}, C_{65}^{*}. Path 1 – 3 – 4 – 2 – 5 - 6.

Tableau 5

		1	2	3	4	5	6	Labeled from
Source	1			5				start
	2	2			3	2		
	3	3			3			1
	4			1				3
	5		2	2			1	
Destination	6				1	4		

Key: Entry in i^{th} row and j^{th} column of each tableau is c_{ij}* blank entries denote zeros

Tableau 4 contains the updated capacities and a summary of the next path search, which used nodes 1, 3, 4, 2, and 5 for labeling. The fifth tableau contains the final updated capacities and path search. Only nodes 1, 3, and 4 can be labeled in this tableau, so the algorithm is completed.

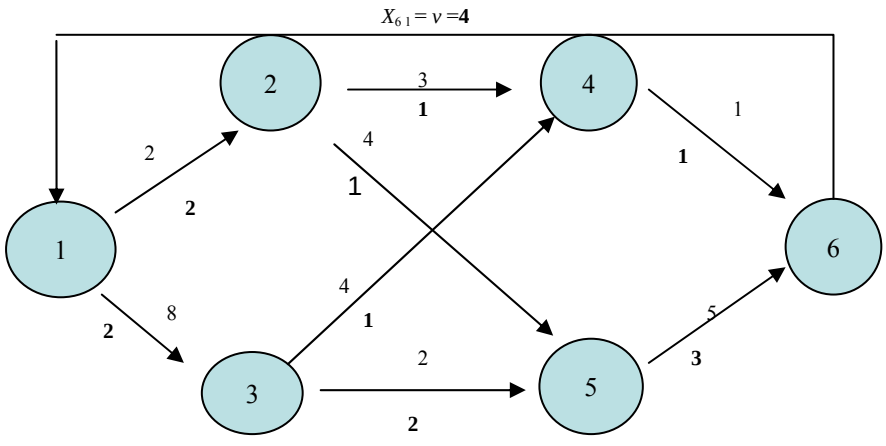


Figure 3.

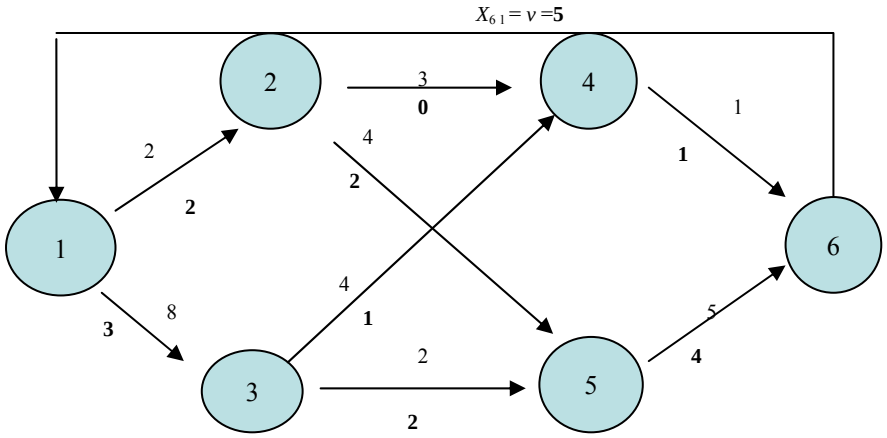


Figure 4.

The flow at any point in the algorithm is obtained by subtracting effective capacities from initial capacities, $c_{ij} - c_{ij}^{*}$ and discarding negative numbers. For Tableaus 1, 2, or 3, the flow is given by Figure1. After making the two indicated flow changes to obtain Tableau 4 and then Tableau 5, the solutions are given in two networks shown in Figures 3 and 4. The optimal solution sends two units along each of the paths 1–2–5–6 and 1–3–5–6 and one unit along 1–3–4–6.By Max-flow Min – cut theorem, the maximum –in-flow is 5 units and the resultant network is given in Figure. 4. Maximum flow can be pushed the network is 5 + 4 =9.

We shall present an algorithm sends flow along all gain augmenting paths simultaneously, using a maximum flow computation the algorithm is given below.

Algorithm 3.5

Procedure max flow (X: network; f: flow; gain function $\gamma : E \rightarrow R$: max - out-flow: real)

{Finds maximum out flow in a given network}

Max -out-flow value: = 0

Set $\gamma = 1 - \xi, where\ 0 < \xi < 1$

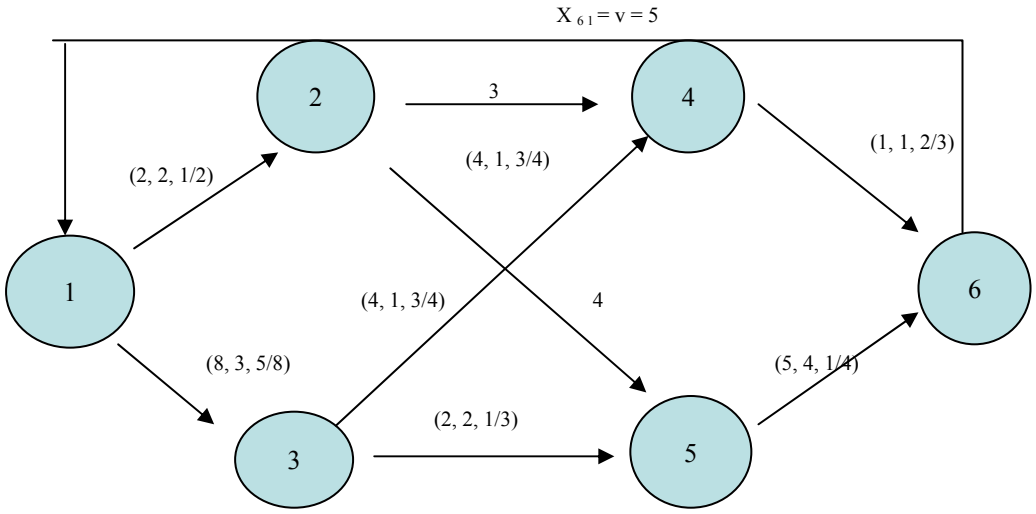
While there exists an augmenting path do

Max – out - flow value: = max – out - flow value + $\gamma$where $\gamma = \sum \gamma(x) + \sum \gamma(y)$

End {while}.

To find maximum out flow

Consider the network with gain function



In this network we have GAP:

F_1: 1 – 3 – 5 – 6 & 1 – 3 – 4 – 6 and F_2: 1 – 2 – 5 - 6.
Let x and y be the gain function of the corresponding paths F_1 and F_2
In F_1, The gain function of the path 1 – 3 – 5 - 6 can be expressed as

$$x \to \frac{3x}{8} = 0.375x,\ \frac{3x}{8} \to \frac{3x}{8} * \frac{2}{3} * \frac{3}{4} = 0.375x * 0.5$$

In the path 1 – 3 - 4– 6, the expression becomes

$$x \to \frac{3x}{8} = 0.375x,\ \frac{3x}{8} \to \frac{3x}{8} * \frac{1}{4} * \frac{1}{3} = 0.375x * 0.0833$$

In F_2, The gain function of the 1 – 2 – 3 – 6 can be expressed as

$$y \to \frac{y}{2} = 0.5y\ ,\ \frac{y}{2} \to \frac{y}{2} * \frac{1}{2} * \frac{3}{4} = 0.5y * 0.375$$

Maximum out flow = sum of the gain function of x + sum of the gain function of y.

Iteration 1

When x = 1, Augment flow along (1, 3, 5, 6)

The gain function $\frac{3x}{8} = 0.375x$

0.375x * 0.5 = 0.1875

Augment flow along (1, 3, 4, 6)

0.375x * 0.0833 = 0.03124

When y = 1, Augment flow along (1, 2, 5, 6)

The gain function $\frac{y}{2} = 0.5y$

0.5y * 0.375 =0.1875

Maximum out flow = 0.1875 + 0.03124 + 0.1875 = 0.4062

Iteration 2

When x = 2, Augment flow along (1, 3, 5, 6)

The gain function $\frac{3x}{8} = 0.375 * 2 = 0.75$

0.75 * 0.5 = 0.375

Augment flow along (1, 3, 4, 6)

0.75 * 0.0833 = 0.06248

When y =2, Augment flow along (1, 2, 5, 6)

The gain function y / 2 = 0.5y = 0.5 * 2 = 1

1* 0.375 = 0.375

Maximum out flow= 0.375 + 0.06248 + 0.375 = 0.8125

Proceeding like this, we will increase the value of x and y, up to x = 8 and add with y = 2. Since the flow value cannot exceed the capacity.

Result: Maximum in flow = 5
Maximum out flow = 2.125

Time complexity

Let V be the number of vertices, V_0 be the number of end vertices and E be the number of edges. Each iteration of the while loop takes $O(E)$ time. Then the above algorithms work with $O(E V_o)$ time. That is, $O(E V)$ time complexity.

IV. APPLICATIONS

In traditional networks, there is an implicit assumption that flow is conserved on every arc. Many practical applications violate this conservation assumption. The gain factors can represent physical transformations of one commodity into a lesser or greater amount of the same commodity. Some examples include: spoilage, theft, evaporation, taxes, seepage, deterioration, interest, or breeding. The gain factors can also model the transformation of one commodity into a different commodity. Some examples include: converting raw materials into finished goods, currency conversion, and machine scheduling. We explain the latter two examples next.

Currency Conversion

We use the currency conversion problem as an example of the types of problems that can be modeled using generalized flows. Later, we will use this problem to gain intuition. In the currency conversion problem, the goal is to take advantage of discrepancies in currency conversion rates. Given certain amount of one currency, say 1000 U.S dollars, the goal is to convert it into the maximum amount of another currency, say French Francs, through a sequence of currency conversions. We assume that limited amounts of currency can be treated without affecting the exchange rates.

Scheduling unrelated parallel machines

As a second example, we consider the problem of scheduling N jobs to run on M unrelated machines. The goal is to schedule all of the jobs by a pre specified time T. Each job must be assigned to exactly one machine. Each machine can process any of the jobs, but at most one job at a time. Machine i requires a pre-specified amount of time P_{ij} to process job j.

REFERENCES

[1] R. K. Abuja, T. L. Magnanti, J. B. Orin, "*Network Flow*," Prentice-Hall, Englewood Cliffs, N. J. (1993)..

[2] J.A. Bandy and U. S. R. Marty,"*Graph theory with applications*," Elsevier SciencePublishing co., Inc., U. S. A. (1976)..

[3] L.R. Ford and D.R. Fulkerson,"*Flows in networks*," Princeton University Press, Princeton, NJ. (1974).

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

[5] H. S. Wolf, *Algorithms and Complexity*, Prentice – Hall International, Inc., U. S. A. (1986).

[6] A. Anto Kinsley and B. uma Maheswari, "Design of algorithms to Generalsed Maximum flow network Problem," International Journal of Advanced Research in Computer Science and Technology(IJARCST 2016) Vol.4,Issue 1 (Jan. – Mar 2016) pp:31- 34.

[7] A. Anto Kinsley and B. uma Maheswari, "A Pre-Flow Push algorithm to Generalsed Maximum flow Problem," International Journal ofEngineering Science and Research Technology(IJESRT) Feb. 2016 pp:547-553.

[8] A. Anto Kinsley and B. uma Maheswari, "Design of algorithms to Maximum flow Problems in tree flow Networks," International Journal of Scienc, Engineering and Technology(IJSETR) Vol.5,Issue 4, April 2016 pp:1015-1018.