International Research Journal of Computer Science (IRJCS)
Issue 05, Volume 4 (May 2017) SPECIAL ISSUE
ISSN: 2393-9842
www.irjcs.com

An Ameliorated Methodology to Compute Maximum Flow in A Flow Network

Dr. Shivanand M. Handigund ,
Dept. of Information Science & Engineering,
Vemana Institute of Technology, Bengaluru 560 034, India,

Ms. Arunaumari B.N.,
Dept. of Computer Science & Engg,
Research Resource Center,
Visvesvaraya Technological University, Belagavi

Dr. Ajeet Chikkamannur,
Dept. of Computer Science & Engineering,
R. L. Jalappa Institute of Technology, Bengaluru 561 203,

Abstract *— The motto of software engineers is to resurrect the information flow of business organization to enable different stakeholders to grasp it in their own perspective views. All au-courant information systems are open system i.e. getting information flow from source (actors) and providing architectonic information flow in the form of software to sink (the same or different actors). The system contains the reticular network of activities procuring appropriate information from source and providing the information in the form of software to the sink. Thus, software development process is analogous to flow problem where we need to obtain maximum information flow from the reticulation of work processes. Ford Fulkerson et al. have developed different techniques to determine the maximum flow from source to sink in running time* $O(VE^2)$*. Jack Edmonds & Richard Karp have further modified the method using breadth first search method reducing the running time to* $O(V^2E)$*. They have not substantiated their claim of success of their techniques. These techniques may narrows down the solution towards success, but still it requires raft of human skills to realize the benefit. This paper attempts to develop a suigeneris method to identify maximum information flow in the flow networks through the matrix computation in running time of* $O(V_bE)$ *where* V_b *is the minimum of boundary nodes either connecting to source or joining to sink. The au-courant flow network methods have not considered any type of data structure though the procedure is iterative. Moreover, the path is identified in enormous running time. There is no automated procedure developed to identify path from source to sink. In our proposed methodology, we have taken only capacity and avoided the determination of augmenting path and residual graph. In the proposed methodology we have attempted to determine the maximum flow with predefined number of iterations through use of matrix data structure.*

Keywords *—Flow network, Breadth First Serach, Depath First Serach, Matrix Data Structure.*

I. INTRODUCTION

The goal of this paper is represented in state-of-the-art form of vision mission and objectives as follows

Vision: To realize the maximum flow problem through an automated process.
Mission: To develop an ameliorated automated process for computation of maximum flow in reticular graph through the use of matrices.

Objectives:

- To determine the maximum flow from source to sink by incrementally identifying the paths from source to sink.
- To effectively utilize the principles of flow networks in automated process.
- To determine the maximum flow through incremental computation using arithmetics on matrices.

A. Motivation

The Ford Fulkerson method contains only broad steps, though it narrows down the arbitrary use of human skills, still success depends on human skills as their steps are not automatically implementable. Their method starts with random sample selection of the flows for each flow capacity edge.

Since, this flow decides the number of iterations to augment for the maximum flow, low values of flow enhances the number of iterations and higher values of flow dampens the risk of not finding a path. This labyrinth needs to be resolved with clairvoyant experience. As a result, the effort cannot be estimated knocking the running time to $O(VE^2)$. Moreover, the complexity of identification of the path is proportional to the number of nodes. The ford Fulkerson method specifies the ends, based on not finding the augmenting path. No data structure is specified for utilization of the method. Edmonds & Karp used the breadth first search (BFS) algorithm to identify the nodes and edges of the path. But unfortunately BFS uses all edges & nodes in all iterations, enhancing superfluously the number of iterations. These sciolistic methods may be of theoretical interest but may not be suitable for information flow within the business process. In the business process there are raft of number of work process and information flows between them. The reticulation of flow network is quite analogical to our software development wherein information flow should be optimally transformed from requirements specification to the development of the software. There is urgent need to determine the maximum flows in optimally limited number of predefined iterations. This may need clairvoyant study of these methods for correcting their deviations to appropriately ossify their suitability for software development. This paper is an attempt in achieving the gal.

B. Litreature Survey

The implementation of [1, 4, 8] Any flow graph is implementable if it is represented by data structure. Since [1, 6, 7] flow graph cannot be represented in data structure (based on matrix form). Hence, method gives some guidelines to identify the maximum flow from source to sink. [2] have identified paths using depth first search (DFS) method but they have not considered the representational of the graph in data structure. At most it serves as theoretical guidelines. [3] used the BFS method [4,5] to identify the paths. The existence of the path is known only at the visit of the sink node. Till penultimate stage the flow is computation is order of V^2. This is only useful if there exists paths of the order $O(V^2)$ otherwise it is waste of computation and memory. Moreover, if an edge in V^2 edges contains minimum flow for the subsequent computation all paths via minimum flow edge may be futile. This estimation is only for acyclic graph and if the graph is cyclic graph then wastage enormously increases. In our method we have used modified DFS algorithm [2, 4, 5] and matrix representation of graph (data structure) to automatically identify the maximum flow. The au-courant flow graphs have not at all considered the principles of the flow constraint which unnecessarily enhances the number of iterations. In the software development any transformation cannot be reversed back. In any of the software development process the ford Fulkerson method is not applicable, its only mathematical interest. In our methodology we attempted to utilize the flow constraint properties effectively in determination of maximum flow. The Push Relabel algorithm [8] developed efficient method which has running time of $O(V^2\sqrt{E})$ to determine maximum flow without using BFS algorithm. However the authors have used the distance concept analogical to distance of a tree from root node. Here, the distance from the sink is considered based on existence of acyclic.

II. PROPOSED METHODOLOGY

The flow graph contains reticular flow of information from source to the reticulation of nodes of the system and from reticulation of system of nodes to the sink. If there exists raft of nodes in the reticulation then the determination of paths should be carried out in an architectonic way. This desiderates the use of matrix to represent the graph to enable the path computation to be made through the data structure of the matrix. Therefore, we have represented graph through the matrix of flows. Each row determines flow from specific node to all other nodes and each column determines the flow from other nodes to a specific node. Thus, V_{ij} represent the flow from node V_i to V_j. To identify the path from source to sink Edmonds-Karp has used BFS which takes running time O(VE) for each node and totalling V^2E for all nodes together. This is the optimized computation with the representation of matrix. We have used depth first search (DFS) algorithm that reduces visiting time to V and processing time to E. So that for each node the running time is $O(V + E) \approx E$. Thus the total running time to applying DFS for all vertices is O (VE). Further, we restrict the application of DFS only to boundary classes between source and system reducing the running time (V_bE) where $|V_b| << |V|$. The algorithm is discussed as follows

Step 1:

Store network flow information (n+1) x (n+1) square matrix (flow matrix) where n is the number of flow receiving nodes including sink. Where i^{th} row represents flow from V_i to all other nodes j, where $j \in [0, n]$. An element V_{ij} where $[i, j] \in [0, n]$ represents flow from node V_i to V_j. Enter the values from the flow graph under consideration assigning rows and columns in the order from source to sink.

Create DFS path matrix of order (4 x n) where n is the number of nodes excluding source node and sink node information at the n^{th} row and n^{th} column as shown below

NODE (F_{0J})	v_1	v_2	v_3	v_4	t
VISITING ORDER (F_{1J})					
FLOW (F_{2J})					
PROCESSING ORDER (F_{3J})					
REMNANT VALUE (F_{4J})					

F_{0j} : the node information with F_{0n} as the sink node
F_{1j} : represents the visiting order of the node V_j
F_{2j} : represents the flow from previously visited node to j^{th} node
F_{3j} : represents processing order of the node F_{2j} through the maximum flow path from source to sink nodes.
F_{4j} : represents the remnant flow quantity if the maximum flow of the path is accounted.

Create a duplicate (n +1) x (n+1) as the mirror of the flow network matrix with all elements zero

Step 2:

Maximum total flow (MF_t) := 0
Flow order (CN) := 0
Maximum flow (MF_p) := 0
V_{ij} := 0
i := 0; j := 1;
Initialize all elements of DFS path matrix to zero

Step 3:

Read V_{ij} , if (V_{ij} = 0); j := j + 1 Until $V_{ij} \neq 0$
$MF_p := V_{ij}$
$F_{1j} := CN + 1$
$F_{2j} := V_{0j}$
Store V_{0j} in the mirror flow network matrix
if ($V_{ij} < MF_p$)
$MF_p := V_{ij}$
i := j
j := k where $k \neq 0$
Repeat above process till $V_{ij} = 0$ and j = n
If (j = n) GOTO step 4
If ($V_{ij} = 0$) ∀ j, GOTO step 5
End procedure when ∀ $V_{ij} = 0$

Step 4:

$F_{3n} := CN + 1$
$F_{4n} := F_{2n} - MF_p$
k := j
j := i
Repeat above process till i = 0
$MF_t := MF_t + MF_p$
GOTO step 6

Step 5:

$F_{3j} := CN + 1$
$F_{4j} := F_{2j}$
k := j
j := i
Initialize the mirror flow matrix to zero
GOTO step 3

Step 6:

If ($V_{ij} \neq 0$) in mirror flow network matrix replace V_{ij} by MF_p
[Flow network matrix] = [Flow Network Matrix] – [Mirror Matrix]
GOTO step 3

This identifies the maximum flow from source to sink of reticular flow of nodes further MF_p identifies the actual flow in the nodes of the path. If the node is participated in two or more paths the sum of all paths minimum flows is the actual flow to the common node.

III. CONCLUSION

The available au-courant methods to determine maximum flow in reticular network of nodes with source and sink have failed to apply appropriate data structure for storage and up gradation of data of their methodologies. In some of the methods BFS and tree structure algorithms have been used to reduce the running time from $O(VE^2)$ to $O(V^2E)$ or $O(V2\sqrt{E})$. We have used matrix data structure where we considered boundary nodes only for initial nodes. This helped us in reducing the running time to $O(V_bE)$, where V_b is minimum of boundary nodes connecting either source or sink. Our algorithm ends in fixed number of iterations. Since $V_b << V$ our methodology is the most efficient, accurate. This may be applied to software development activities.

ACKNOWLEDGMENT

The Words are insufficient to express our deep sense of gratitude to Dr. D. B. Phatak, Professor of Dept. of Computer Science & Engineering IIT Bombay, for his inspiration.

REFERENCES

[1]. G. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein (2009). "26.2". Introduction to Algorithms (third ed.). MIT Press. pp. 727–730. ISBN 978-0-262-03384-8.
[2]. Dr. S. M. Handigund and Swetha Bhat, "Automated Methodologies for the Design of Flow Diagram for Development and Maintenance Activities" (paper no.ICETS-SI-2010-46) has been published in Springer E-business Technology and Strategy Communications in Computer and Information Science, 2010, Volume 113, 93-104, DOI: 10.1007/978-3-642-16397-5_8.
[3]. Edmonds Jack, Karp, Richard M. (1972). "Theoretical improvements in algorithmic efficiency for network flow problems". Journal of the ACM. Association for Computing Machinery. 19 (2): 248–264. doi:10.1145/321694.321699.
[4]. Even Shimon, "Graph Algorithms (2nd edition), Cambridge University Press, (2011), pp. 46–48, ISBN 978-0-521-73653-4.
[5]. Aziz Adnan, Prakash, Amit (2010). "4. Algorithms on Graphs". *Algorithms for Interviews*. p. 144. ISBN 1453792996.
[6]. Dinic, E. A. (1970). "Algorithm for solution of a problem of maximum flow in a network with power estimation". Soviet Math. Doklady. Doklady. 11: 1277–1280.
[7]. Yefim Dinitz. "Dinitz's Algorithm: The Original Version and Even's Version" (PDF).
[8]. Push–relabel maximum flow algorithm from wikipedia,
[9]. https://en.wikipedia.org/wiki/Push%E2%80%93relabel_maximum_flow_algorithm

APPENDIX

CASE STUDY OF FORD FULKERSON METHOD

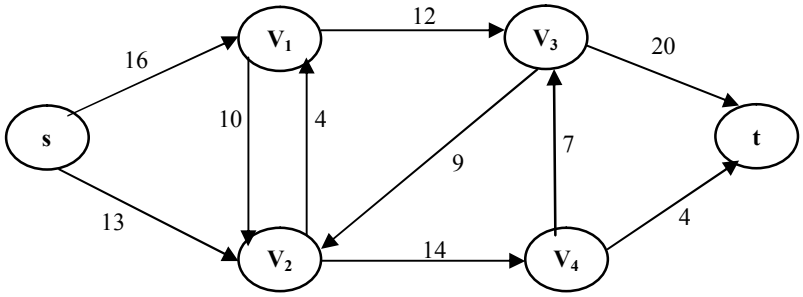

TABLE 1: ORIGINAL FLOW NETWORK MATRIX

	S	v_1	v_2	v_3	v_4	t
S	0	16	13	0	0	0
v_1	0	0	10	12	0	0
v_2	0	4	0	0	14	0
v_3	0	0	9	0	0	20
v_4	0	0	0	7	0	4
T	0	0	0	0	0	0

TABLE 2 : DFS PATH MATRIX

NODE	v_1	v_2	v_3	v_4	t
VISITING ORDER	1		2		3
FLOW	16		12		20
PROCESSING ORDER	6		5		4
REMNANT VALUE	4		0		8

	S	v_1	v_2	v_3	v_4	T
S	0	~~16~~ **4**	13	0	0	0
v_1	0 **12**	0	10	~~12~~ **0**	0	0
v_2	0	4	0	0	14	0
v_3	0	0 **12**	9	0	0	~~20~~ **8**
v_4	0	0	0	7	0	4
T	0	0	0	**12**	0	0

Iteration 1

Iteration 1's Mirror flow matrix

	s	v_1	v_2	v_3	v_4	t
s	0	12	0	0	0	0
v_1	0	0	0	12	0	0
v_2	0	0	0	0	0	0
v_3	0	0	0	0	0	12
v_4	0	0	0	0	0	0
t	0	0	0	0	0	0

Iteration 2:

	S	v_1	v_2	v_3	v_4	t
S	0	4	~~13~~ **9**	0	0	0
v_1	12	0	10	0	0	0
v_2	0+**4**	4	0	0	~~14~~ **10**	0
v_3	0	0	9	0	0	8
v_4	0	0+**4**	0	7	0	~~4~~ **0**
T	0	0	0	**12**	**4**	0

Iteration 2's Mirror flow matrix

	S	v_1	v_2	v_3	v_4	t
s	0	0	4	0	0	0
v_1	0	0	0	0	0	0
v_2	0	0	0	0	4	0
v_3	0	0	0	0	0	0
v_4	0	0	0	0	0	4
t	0	0	0	0	0	0

Iteration 3

	S	v_1	v_2	v_3	v_4	T
S	0	4	~~9~~ **2**	0	0	0
v_1	0	0	10	0	0	0
v_2	4+7	4	0	0	~~10~~ **3**	0
v_3	0	0	9	0	7	~~8~~ **1**
v_4	0	4	7	~~7~~ **0 7**	0	0
T	0	0	0	12 +7	4	0

Iteration 3's Mirror flow matrix

	S	v_1	v_2	v_3	v_4	t
s	0	0	7	0	0	0
v_1	0	0	0	0	0	0
v_2	0	0	0	0	7	0
v_3	0	0	0	0	0	7
v_4	0	0	0	7	0	0
t	0	0	0	0	0	0

Iteration 4:

	S	v_1	v_2	v_3	v_4	t
S	0	4	**2**	0	0	0
v_1	12	0	10	0	0	0
v_2	11	4	0	0	**3**	0
v_3	0	0	9	0	0	**1**
v_4	0	11	0	**0**	0	0
T	0	0	0	12+7	4	0

Maximum total flow 12 + 4+ 7 = 23