# Graphical Maximal Flow Analytics Tool

Gokul Lokhande
*Department of Computer Engineering*
*Sinhgad College Of Engineering, Pune, Maharashtra, India*

Devashree Davate
*Department of Computer Engineering*
*Sinhgad College Of Engineering, Pune, Maharashtra, India*

**Abstract - Previously known efficient maximum-flow algorithm of Ford-Fulkerson worked by finding augmenting paths. In this paper, a simple and systematic method for finding maximal flow in a Job scheduling application by Edmonds-Karp algorithm has been introduced. The algorithm has time complexity of $O(EV^2)$, with a simple and intuitive analysis better than Ford- Fulkerson's algorithm.**

**Keywords – Maximal flow, Edmonds-Karp algorithm**

## I. INTRODUCTION

The simplest form that the statement, 'The Standard Maximal Flow Problem' could describe will be as 'A list of underground pipes is given, with different flow-capacities. These pipes are connected at their endpoints. What is the maximum amount of gas that can route from a given starting point to a given ending point?'
The 'Maximal Flow Analytics Tool' is an application that helps us to solve the problem of job assignment. To obtain maximum throughput, use of this tool can be the optimal choice.

### *1.1 Background–*

Network flows are of growing interest now-a-days from point of view of both application as well as theory. The topic of network flow has applications in such diverse fields such as engineering, management science, computer science, urban traffic, communications, and economics to name but a few. Maximal flow involves finding a feasible flow. This flow will be through a single-source, single-sink flow.

### *1.2 Scope–*

We are developing Graphical maximal flow analytics tool that can be used for Job assignment problem wherein a graphical interface is provided and we have an underline implementation of Edmonds-Karp algorithm. The input parameters will be the number of workers and number of jobs. The maximal flow will be calculated using Edmond-Karp algorithm. And final job assigned graph will be displayed.

## II. PROPOSED ALGORITHM

### *2.1 Edmonds-Karp algorithm–*

The input parameters to the algorithm are Capacity matrix, Neighbour lists, Source, Sink and output parameters are Value of maximum flow, a matrix which will give a legal flow. This legal flow will should be of maximum value. Initially maximum flow is initialized to 0. Then a loop is started with exiting condition as there doesn't exist capacity of path found. BFS is run to find out the capacity of path found and to update parent table. Capacity of path found is added to the maximum flow if it is not zero. A node V is chosen and if V is not equal to the sink node then backtrack search and write flow i.e. update flow matrix.
The algorithm uses a BFS (Breadth First Search) technique. It works as follows:
The input parameters are C, E, s, t, F and output parameters are Capacity of path found and Parent table. Initialize the Parent table. Push source vertex into the Queue. Then while Queue size is greater than zero then pop first vertex and check its neighbour vertices i.e. if there exists a capacity and neighbour vertex is not seen in the search before. If there exists such neighbour vertex v then make entry of u in the parent table i.e. P[v]=u. Then take the shortest path

i.e. choose minimum between m and residual capacity(C[u][v]-F[u][v]).Now check if v is sink then return the m and Parent table else add v to Queue.

**Algorithm Edmond-Karp:**

```
1.  Input:
2.  C[1..n, 1..n] (Capacity matrix)
3.  E[1..n, 1..?] (Neighbour lists)
4.  s           (Source)
5.  t           (Sink)
6.  Output:
7.  f         (Value of maximum flow)
8.  F       (A matrix giving a legal flow with the maximum value)
9.  f := 0 (Initial flow is zero)
10. F := array(1..n, 1..n) (Residual capacity from u to v is C[u,v] - F[u,v])
11. forever
12. m, P := BreadthFirstSearch(C, E, s, t, F)
13. if m = 0
14. break
15. f := f + m
16. (Backtrack search, and write flow)
17. v := t
18. while v ? s
19. u := P[v]
20. F[u,v] := F[u,v] + m
21. F[v,u] := F[v,u] - m
22. v := u
23. return (f, F)
```

**Algorithm BreadthFirstSearch:**

```
1.  input:
2.  C, E, s, t, F
3.  output:
4.  M[t]         (Capacity of path found)
5.  P            (Parent table)
6.  P := array(1..n)
7.  for u in 1..n
8.  P[u] := -1
9.  P[s] := -2 (make sure source is not rediscovered)
10. M := array(1..n) (Capacity of found path to node)
11. M[s] := ∞
12. Q := queue()
13. Q.push(s)
14. while Q.size() > 0
15. u := Q.pop()
16. for v in E[u]
17. (If there is available capacity, and v is not seen before in search)
18. if C[u,v] - F[u,v] > 0 and P[v] = -1
19. P[v] := u
20. M[v] := min(M[u], C[u,v] - F[u,v])
21. if v ? t
22. Q.push(v)
23. else
24. return M[t], P
25. return 0, P
```

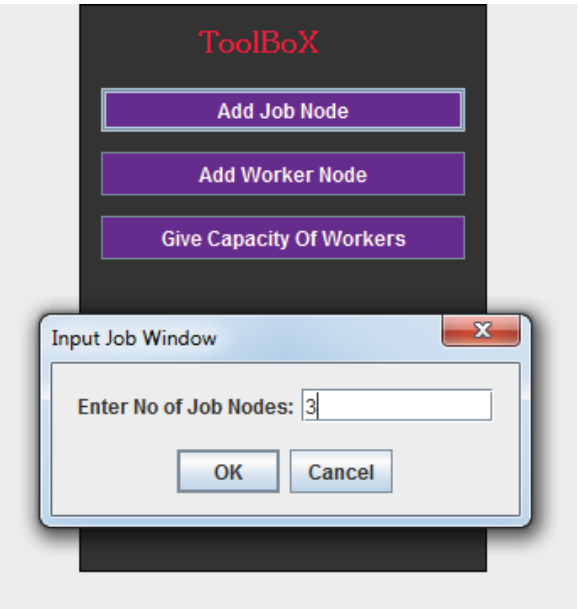
*List of Symbols:*

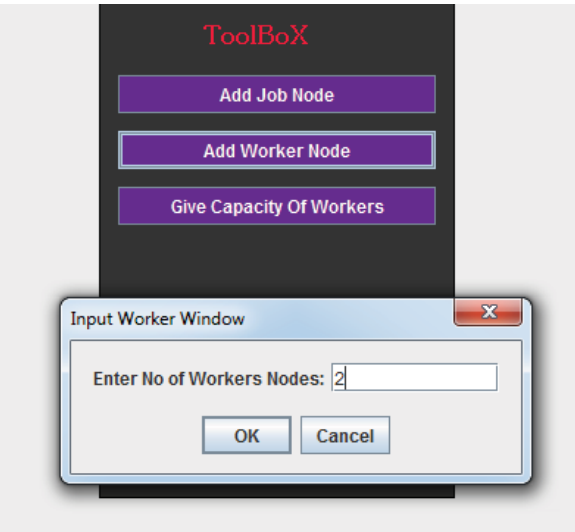
C[n][n] - Capacity matrix
E[n][n] - Neighbour lists
s - Source
t - Sink
f - Value of maximum flow
F - A matrix giving a legal flow with the maximum value
M[t] - Capacity of path found
P[] - Parent table

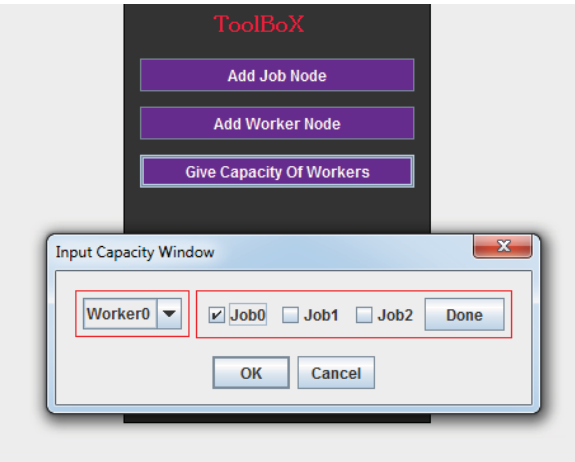
2.2 *Graphical Interface–*
User should be able to give input using GUI and final job assigned graph will be shown on the GUI.

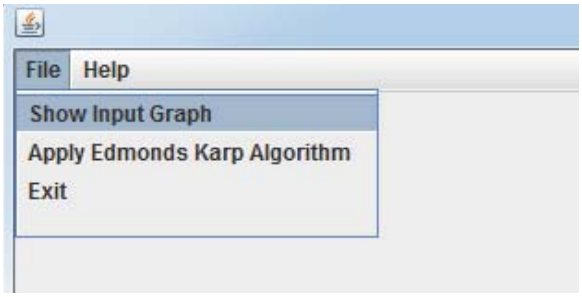
2.3 *Maximal Flow Calculation–*
The analyst will give input as number of workers, number of jobs and the capacities of workers. A graph will be created from this input. From the drawn graph maximal flow can be calculated. For calculation of maximal flow we will be using Edmonds-Karp algorithm for finding augmenting path. The path found must be a shortest path that has available capacity. This can be found by a breadth-first search, as we let edges have unit length. Another property of this algorithm is that there is a monotonic increase in the length of the shortest augmenting path.

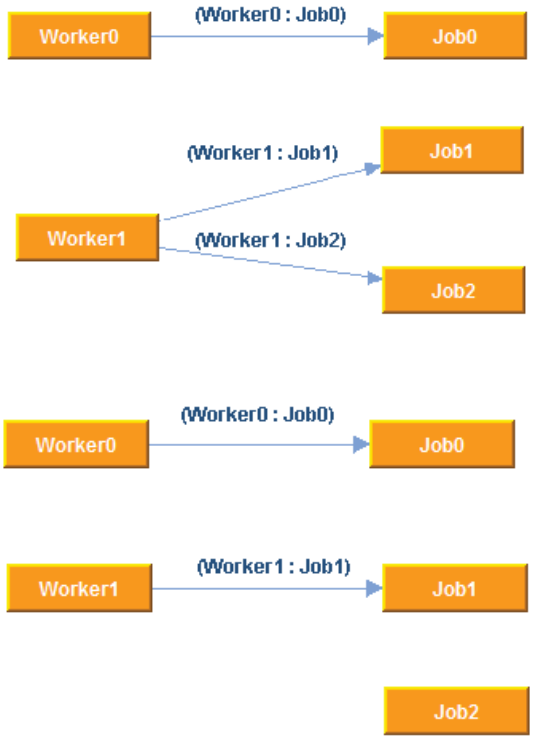
## III. Experiment and Result




ToolBoX
Add Job Node
Add Worker Node
Give Capacity Of Workers
Input Worker Window
Enter No of Workers Nodes: 2
OK
Cancel


ToolBoX
Add Job Node
Add Worker Node
Give Capacity Of Workers
Input Capacity Window
Worker0
Job0
Job1
Job2
Done
OK
Cancel


File
Help
Show Input Graph
Apply Edmonds Karp Algorithm
Exit




```
run:
32Capacity Matrix:

 1 0 0
 0 1 1Matrix IS

 0 1 1 0 0 0 0
 0 0 0 1 0 0 0
 0 0 0 0 1 1 0
 0 0 0 0 0 0 1
 0 0 0 0 0 0 1
 0 0 0 0 0 0 1
 0 0 0 0 0 0 0 0Flow Matrix is

 0 1 1 0 0 0 0
 0 0 0 1 0 0 0
 0 0 0 0 1 0 0
 0 0 0 0 0 0 1
 0 0 0 0 0 0 1
 0 0 0 0 0 0 0
 0 0 0 0 0 0 0
Job assigned Matrix is

 1 0 0
 0 1 0Maximum Flow is:2
```

## IV.CONCLUSION

From the study it is proved that Edmonds-Karp algorithm is better than Ford-Fulkerson's algorithm as in Edmonds-Karp algorithm, search order when finding the augmenting path is defined. Thus it has time complexity of O ($EV^2$) better than the Ford-Fulkerson algorithm which has time complexity of O (max flow*E).

## REFERENCES

[1] Andrew V. Goldsberg and Robert E. Tarjan, "A New Algorithm to Maximal Flow Problem", Journal of the Association for Computing Machinery. Vol. 35 No. 4. October 1988, pp. 921-940.

[2] Farhad Shahrokhi and D.W.Matula, "Maximal Concurrent Flow Problem", Journal of the Association for Computing Machinery. Vol. 37, No. 2. April 1990, pp.318-334.

[3] Zvi Galil, "A New Algorithm for Maximal Flow Problem", Department of Mathematical Sciences Computer-Science Division, Tel-Aviv University Ramat-Aviv, Tel-Aviv, Israel.

[4] Thomas H. Cormen, Ronald L. Rivest, "Introduction to Algorithms".