Theoretical Computer Science 14 (1981) 103–111
North-Holland Publishing Company

ON THE THEORETICAL EFFICIENCY OF VARIOUS NETWORK FLOW ALGORITHMS

Zvi GALIL*
Department of Mathematical Sciences, Computer Science Division, Tel-Aviv University, Israel

Communicated by M. Paterson
Received December 1978
Revised March 1980

Abstract. The time bounds of various algorithms for finding the maximal flow in networks are shown to be tight by constructing *one* parametrized family of network flow problems and showing that the algorithms achieve the corresponding bounds on inputs from this family.

1. Introduction

One of the well-known problems in combinatorial optimization is the problem of finding the maximal flow in a given network (max-flow in short). The problem was posed and solved by Ford and Fulkerson [8]. (See also [7].) The interesting history of the problem is summarized in Table 1, that gives the upper bounds of the time and space complexities of the various algorithms in terms of the number of vertices V and the number of edges E in the network.

The runtime of the first algorithm cannot be bounded above by a function of V and E. This fact is well known (see for example [9] for more details) and therefore we will deal with the other algorithms only. The last algorithm by Malhotra, Pramodh Kumar and Maheshwari [13] does not improve algorithms number 3 to 5. However, it is very simple and probably is the best when the graph is very dense ($E \approx V^2$).

In this paper we construct *one* parameterized family of network flow problems $\{P_{l,m,n} \mid l, m, n > 0\}$. For every $V > 0$ and every E, $V \leq E \leq V^2$, we can fix l, m, n so that the network of $P_{l,m,n}$ will have no more than V vertices and no more than E edges and each algorithm in Table 1 will require time and space proportional to the corresponding entries in Table 1 for solving $P_{l,m,n}$. With regard to space we will only show that Karzanov's algorithm requires space proportional to V^2. The fact that the others use linear space is straightforward.

* Part of the work reported in this paper was done while the author was visiting the Mathematical Sciences Department at the IBM Thomas J. Watson Research Center. At Tel-Aviv University the work of the author was supported in part by the Bat-Sheva Fund.

Table 1
The various solutions

	Solutions	Time	Space
0	Ford and Fulkerson (1956)	—	E
1	Edmonds and Karp (1969)	E^2V	E
2	Dinic (1970)	EV^2	E
3	Karzanov (1973)	V^3	V^2
4	Cherkasky (1976)	$V^2\sqrt{E}$	E
5	Galil (1978)	$V^{5/3}E^{2/3}$	E
6	The three Indians (1978)	V^3	E

There are several known partial results concerning the bounds in Table 1. Zadeh [13] showed that the bound of Edmonds and Karp's algorithm is tight in case $E = V^2$ and Even and Tarjan [6] noted that the same examples imply that the bound of Dinic's algorithm is tight in case $E = V^2$. However Zadeh's examples seem to require a number of edges proportional to V^2. Baratz [1] showed that the bound on Karzanov's algorithm is tight. However, his examples require only $O(V^2)$ steps by the three Indians' algorithm. Also, his examples require time proportional to V^3 by Karzanov's algorithm only if the edges are scanned in a certain order. If we use the heuristic of scanning edges with smaller capacities first, then his examples require only $O(V^2)$ steps. A. Itai [11] has constructed another example that shows that the bound of Karzanov's algorithm is tight.

We assume that the reader is familiar with the various algorithms and will not discuss them here. The last five algorithms in Table 1 use Dinic's approach. In each phase Dinic constructs a layered network and solves a subproblem on it. The four improvements on Dinic show how to solve the subproblem faster. Although Edmonds and Karp do not construct the layered network one can implicitly find it in their algorithm. We define a phase in Edmonds and Karp's algorithm to be the period of time during which the shortest flow augmenting path is of the same length. Consequently, there is an obvious correspondence between a phase in Dinic's algorithm and a phase in Edmonds and Karp's algorithm.

The structure of the paper is the following: In Section 2 we construct $G_{l,m,n}$ the network of $P_{l,m,n}$, and then describe for a given V, E, how to choose l, m, n so that the number of vertices [edges] in $P_{l,m,n}$ is at most V [E] and the number of phases required by Dinic's algorithm (or Edmonds and Karp's algorithm) is proportional to V. In Section 3 we explain why each algorithm requires time proportional to the corresponding entry in Table 1 divided by V to complete one phase, and thus prove that the bounds in Table 1 are tight.

2. The networks $G_{l,m,n}$

$G_{l,m,n}$ has three sections: the s-section (the section with endpoints s, p, q' in Fig. 1), the t-section (the one with endpoints t, p', q) and the middle section (the rest).

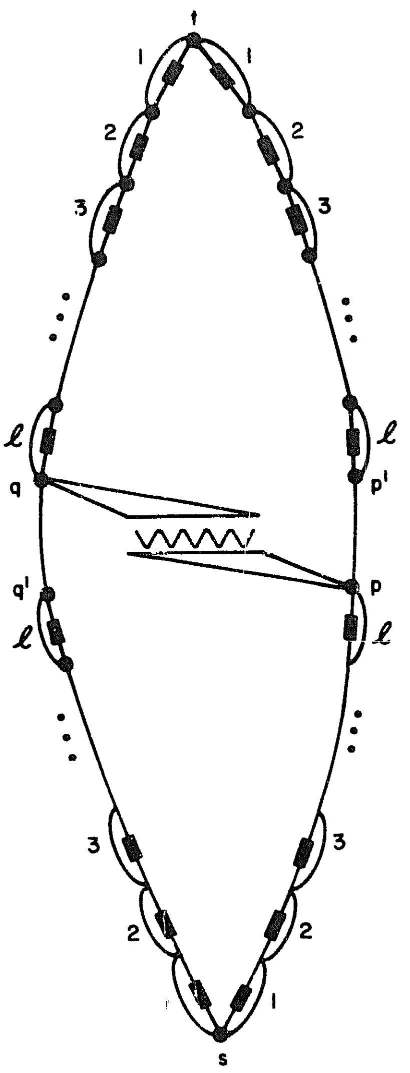

Fig. 1.

The middle section is described in Fig. 2. It consists of two sets of vertices S_1 and S_2 with m vertices each. Each vertex in S_1 [S_2] is connected to n vertices in S_2 [S_1]. These nm edges are called *bottlenecks* and their capacities are 1. In addition there are ten more vertices on two simple paths (from q' to q and from p to p'). Also, p [q] is connected to all the vertices in S_1 [S_2]. All the edges in the middle section except the bottlenecks have infinite (or large enough) capacities. All together there are $2m+10$ vertices and $nm+2m+8$ edges in the middle section.

The s-section [the symmetric t-section] is composed of $2l$ *units*, l of them between s and p [p' and t] and l of them between s and q' [q and t]. A unit is described in more detail in Fig. 3. All the edges in the unit except the marked one have infinite capacity. The marked one is called the *bottleneck of the unit* and has capacity *imn* if it is the ith unit from s [t]. Such a unit is called a *unit of type i* and there are four units of type i for $1 \leq i \leq l$. We distinguish between them by referring to the upper [lower] left [right]

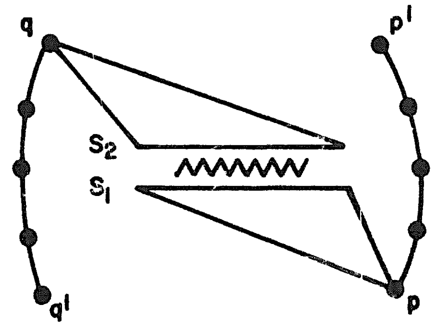

Fig. 2.

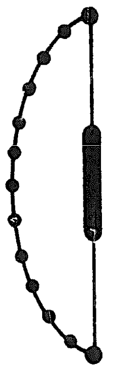

Fig. 3.

type i unit in Fig. 1. The part of a unit that includes its bottleneck is called the *shorter part* and is a simple path of length 3. The other part is called the *longer part* and is a simple path of length 12.

The number of vertices in the s-section [t-section] is $28l$ and the number of edges is $30l$. Consequently the number of vertices in $G_{l,m,n}$: $V_{l,m,n} = 56l+2m+10$ and the number of edges is $E_{l,m,n} = mn+60l+2m+8$.

We first describe what happens in each phase. We assume inductively that after $2i$ phases $0 \leq i < l$, the flow is $2imn$, the bottlenecks in all units of type j, $j \leq i$, are saturated, the flow in the bottlenecks of the other units is imn, and there is no flow in the bottlenecks of the middle section. Notice that the claim holds when $i = 0$. We will show that after two more phases the claim will hold for $i+1$.

The layered network at the beginning of the $(2i+1)^{\text{st}}$ phase is described in Fig. 4. Units of type larger than i will be represented in it by their shorter part. Units of type not larger than i will be represented by their longer part. (There is one exception that will be explained below.)

Except for the shorter parts and longer parts of some units that do not appear in the layered network two more edges will be missing and they are pointed at by arrows numbered one and two in Fig. 4. The first is the last edge on the path from q' to q. This is because the paths $p \to S_1 \to S_2 \to q$ are of length 3, the path $q \to {}^*q$ is of length 4, and

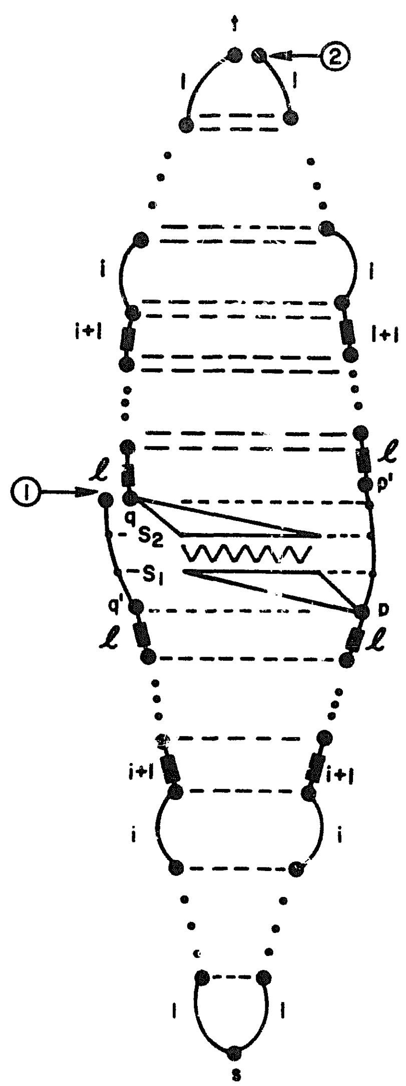

Fig. 4.

p and q' are in the same layer. The second missing edge is the last edge in the longer part of the upper right type 1 unit. (This is the exception mentioned above.) The reason for this is that the paths $p \to S_1 \to S_2 \to q \to^* t$ are shorter by one than the path $p \to^* p' \to^* t$. (Their length are $12i + 3(l-i) + 3$ and $12i + 3(l-i) + 4$ resp.)

During the $(2i+1)$st phase mn of units of flow will be added saturating the bottlenecks of the middle section, and the bottlenecks of the lower right and upper left units of type $i+1$.

At the beginning of the $(2i+2)^{\text{nd}}$ phase, the layered network will look like the one in Fig. 5. This time the units of type j, $j \geq i$, and the lower right and upper left units of type $i+1$ will be represented by their longer part and the other units by their shorter part. There are two exceptions that will be explained below. Except for the parts of some units that do not appear in the layered network, three more edges will be missing and they are pointed at by arrows numbered one and two in Fig. 5. The first is

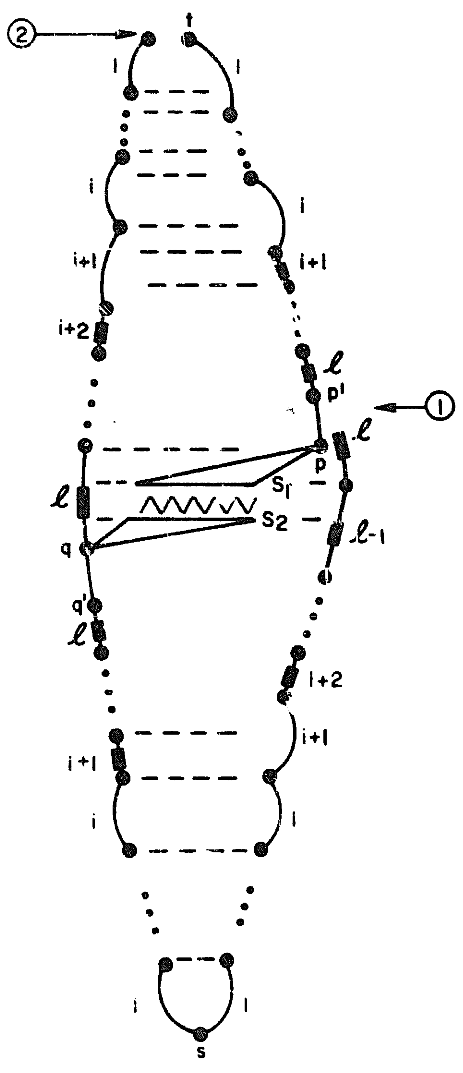

Fig. 5.

the last edge on the short part of the unit of type l that enters p. This is because the distance from s to p going only on the right side is $R - 12(i+1)+3(l-i-1)$, while the distance going to p via $q' \to {}^*q \to S_2 \to S_1 \to p$ is $12i+3(l-i)+7 = R-2$. The other two are the last edges on the longer part of the upper left type 1 unit. Again, the reason is that the length of the path from q to t on the left is R while the length of the path $q \to S_2 \to S_1 \to p \to {}^*t$ is $R-2$. Consequently mn additional flow units will be pushed during the $(2i+2)^{\text{nd}}$ phase. The bottlenecks of the middle section will become flowless and the bottlenecks of the lower left and upper right type $(i+1)$ units will become saturated. As a result the flow after $2i+2$ phases is $(2i+2)mn$, the bottlenecks of all units of type j, $j \leq i+1$, are saturated, flow in the bottlenecks of other units is $(i+1)mn$, and the flow in the bottlenecks of the middle section is zero. This completes the proof of the inductive claim for $i+1$.

We can conclude that the number of phases is $2l$, and that the layered graph in *every* stage looks like the one in Fig. 6 if we ignore the vertices from which t is not reachable.

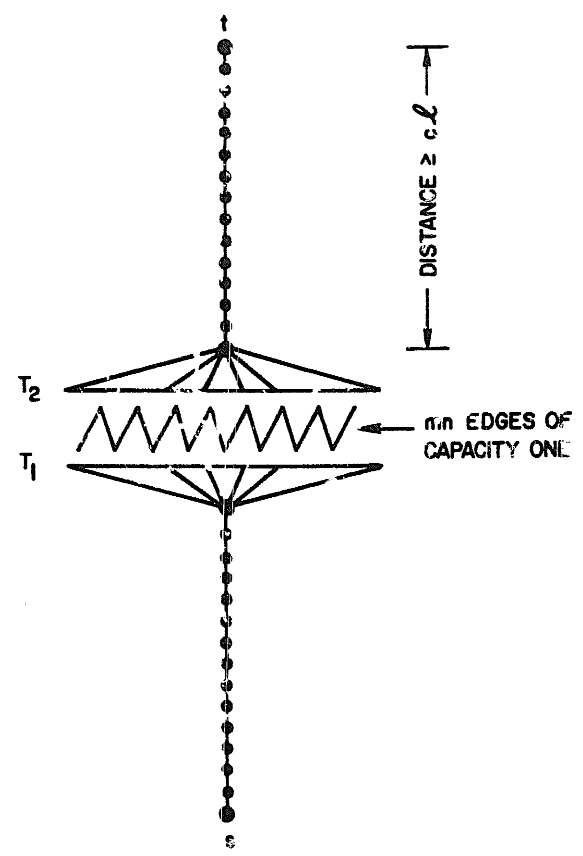

Fig. 6.

Given V and E such that $120 \leq V \leq E \leq V^2$ choose $m = \lfloor \frac{1}{8}V \rfloor$, $l = \lfloor \frac{1}{120}V \rfloor$ and $n = 2\lfloor E/V \rfloor$. It follows that $V_{l,m,n} = 2m + 56l + 10 \leq \frac{1}{4}V + \frac{1}{2}V + 10 \leq V$ and $E_{l,m,n} = nm + 60l + 2m + 8 \leq \frac{1}{8}E + \frac{1}{2}V + \frac{1}{4}V + 8 \leq E$. Also, the number of phases is $2l = 2\lfloor \frac{1}{56}V \rfloor = \Omega(V)$. The assumption that $V \geq 120$ does not limit generality because if $V \leq 120$ and $V \leq E \leq V^2$ all the bounds in Table 1 are shown to be tight by a proper choice of the constant.

3. The behavior of the various algorithms

As was noted in the introduction, Edmonds and Karp's algorithm does not construct the layered network. However, in any phase, any time it finds a flow augmenting path (f.a.p.) it traverses at least the edges of the layered network that have not been deleted yet at this point in that phase. The layered network at the beginning of each phase is described in Fig. 6. The algorithm will discover mn f.a.p.'s in each phase. Each f.a.p. will increase the flow and will lead to the deletion of the corresponding saturated bottleneck. Consequently, the number of steps by Edmonds and Karp's algorithm during one phase, is $\Omega((mn)^2) = \Omega(E^2)$.

Dinic's algorithm will require only a number of steps proportional to the length of the layered network to discover each f.a.p. So each of the $mn(=\Omega(E))$ f.a.p.'s will require $\Omega(l)(=\Omega(V))$ steps. Consequently the number of steps by Dinic's algorithm during one phase is $\Omega(EV)$.

Karzanov's algorithm will bring mn units of flow to v_1 – the first vertex of T_1. It will be able to push forward n units of flow through the n bottlenecks leaving v_1. These units of flow will be pushed to t. Then the excess of flow on v_1 will be balanced and be transferred to v_2 – the second vertex of T_1 and so on. So in one phase n units of flow will be pushed m times from T_2 to t. Each of these $m(=\Omega(V))$ pushes requires $\Omega(l)(=\Omega(V))$ steps. Consequently the number of steps by Karzanov's algorithm during one phase is $\Omega(V^2)$. Moreover, each of the $\Omega(l)(=\Omega(V))$ vertices on the way from T_2 to t has a stack that records the m $(=\Omega(V))$ flow increments due to the m pushes of flow. Therefore, Karzanov's algorithm requires $\Omega(V^2)$ space.

Cherkasky's [Galil's] algorithm selects some of the layers to be special layers. The special layers are chosen in such a way that $Ex = VE^{1/2}[Ex = (VE)^{2/3}]$, where x is a bound on the number of layers between two special layers. In the case of the layered networks of Fig. 6 the special layers can be chosen to be in equal distance from one another. (The only exception is that neither T_1 nor T_2 can be a special layer.) A superlayer consists of two successive special layers and the vertices and edges in between them. Consider the superlayer that contains the bottlenecks. For pushing the flow through this superlayer Cherkasky's algorithm requires x steps for pushing each unit of flow through one of the bottlenecks. Consequently it requires $\Omega(nmx) = \Omega(Ex) = \Omega(VE^{1/2})$ steps per phase. Galil's algorithm constructs in each superlayer a forest with leaves in one special layer and roots in the other. However, since all special layers in our case contain one vertex only, each forest consists of one big edge. The 'forest' which corresponds to the superlayer with the bottlenecks will change nm times. Each time a unit of flow is pushed through it the unique big edge will become saturated, and the 'forest' will be reconstructed after $\Omega(x)$ steps. Consequently, Galil's algorithm requires $\Omega(mnx) = \Omega(Ex) = \Omega((VE)^{2/3})$ steps per phase.

The three Indians' algorithm finds in the layered network the vertex with minimal flow potential α. In our case this will always be one of the vertices in $T_1 \cup T_2$. The algorithm pushes α units of flow from this vertex to t and from it back to s and then deletes this vertex. As a result at least $m(=\Omega(V))$ vertices will be deleted in this way. Each push of flow requires at least $\Omega(l) = \Omega(V)$ steps. Consequently, the three Indians' algorithm requires $\Omega(V^2)$ steps per phase.

Since the number of phases is $2l = \Omega(V)$ we can conclude that the number of steps that the algorithms above need to solve $P_{l,m,n}$ is proportional to the corresponding entries in Table 1.

4. Conclusion

We have constructed a parametrized family of network flow problems $\{P_{l,m,n} \mid l, m, n > 0\}$ and showed how for given V and E, $V \leq E \leq V^2$, we can fix l, m, n

so that the network obtained has at most V vertices and E edges, and requires time proportional to the corresponding entry in Table 1 when processed by the various algorithms. Thus we have proved that the time bounds in Table 1 are tight. We have constructed the simplest possible network problems. Consequently, one can modify the algorithms by adding to them heuristics that take advantage of the simple form of the layered networks that arise. We have two such heuristics: One takes advantage of the fact that all layers except two have exactly one vertex. The other makes use of the fact that most of the edges of the layered networks that arise appear between two layers. The modified algorithms no longer require times proportional to those in Table 1.

However, we were able in both cases to modify $P_{l,m,n}$ (by slightly complicating them) in such a way that the additional heuristics do not help.

Recently, the author together with Naamad discovered an $O(EV \log^2 V)$ algorithm for the max-flow problem [10]. The family of networks described above, requires only $O(EV \log V)$ time by this algorithm. More complicated examples are needed to establish the tightness of the $O(EV \log^2 V)$ time bound. These examples are described in [10].

References

[1] A.E. Baratz, Construction and analysis of network flow problem which forces Karzanov algorithm to $O(n^3)$ running time, MIT, LCS, TM -83, (1977).

[2] B.V. Cherkasky, Algorithm of construction of maximal flow in networks with complexity of $O(V^2\sqrt{E})$ operations (in Russian), *Math. Methods Solution Econ. Problems* **7** (1977) 117-125.

[3] E.A. Dinic., Algorithm for solution of a problem of maximal flow in a network with power estimation, *Soviet Math. Dokl.* **11** (1970) 1277-1280.

[4] J. Edmonds and R. M. Karp, Theoretical improvement in algorithmic efficiency for network flow problems, *J.ACM* **19** (2) (1972) 248-264.

[5] S. Even, The max-flow algorithm of Dinic and Karzanov: An exposition, MIT, LCS, TM-80 (1976).

[6] S. Even and R. E. Tarjan, Network flow and testing graph connectivity, *SIAM J. Comput.* **4** (1975) 507-518.

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

[8] L.R. Ford and D.R. Fulkerson, Maximal flows through a network, *I.R.E. Trans. information Theory* (2) (1956) 117-119.

[9] Z. Galil, A new algorithm for the maximal flow problem, *Proc. 19th IEEE Symposium on Foundations of Computer Science*, Ann-Arbor, MI (1978) 231-245; to appear in *Acta Informat.* as "An $O(E^{2/3}V^{5/3})$ algorithm for the maximal flow problem".

[10] Z. Galil and A. Naamad, Network flow and generalized path compression, *Proc. 11th Annual ACM Symposium on Theory of Computing* (1979) 13-26; to appear in *J. Comput. System Sci.* as "An $O(EV\log^2 V)$ algorithm for the maximal flow problem".

[11] A. Itai, private communication (1978).

[12] A.V. Karzanov, Determining the maximal flow in a network by the method of preflows, *Soviet Math. Dokl.* **15** (1974) 434-437.

[13] V.M. Malhotra, M. Pramodh Kumar and S.N. Maheshwari, An $O(V^3)$algorithm for finding the maximum flows in networks, *Information Processing Lett.* **7**, (6) (1978) 277-278.

[14] N. Zadeh, Theoretical efficiency of the Edmonds-Karp algorithm for computing maximal flows, *J.ACM* **19** (1) (1972) 184-192.