Computing Maximum Flows in Undirected Planar Networks with Both Edge and Vertex Capacities

Xianchao Zhang[1], Weifa Liang[2], and Guoliang Chen[3]

[1] School of Software, Dalian University of Technology
Dalian, China, 116620
[2] Department of Computer Science, Australian National University
Canberra, ACT 0200, Australia
[3] Department of Computer, University of Science and Technology of China
Hefei, China, 230027
xczhang@dlut.edu.cn, wliang@cs.anu.edu.au, glchen@ustc.edu.cn

Abstract. We study the maximum flow problem in an undirected planar network with both edge and vertex capacities (EVC-network). A previous study reduces the minimum cut problem in an undirected planar EVC-network to the minimum edge-cut problem in another planar network with edge capacity only (EC-network), thus the minimum-cut or the maximum flow value can be computed in $O(n \log n)$ time. Based on this reduction, in this paper we devise an $O(n \log n)$ time algorithm for computing the maximum flow in an undirected general planar EVC-network and an $O(n)$ time algorithm for computing the maximum flow in an undirected (s,t)-planar EVC-network. As a result, the maximum flow problem in undirected planar EVC-networks is as easy as the problem in undirected planar EC-networks in terms of computational complexity.

1 Introduction

The maximum flow problem in a flow network with both edge and vertex capacities (EVC-network) is to find a flow between a pair of vertices such that the value of the flow is maximized. This is a classical combinatorial optimization problem with a wide variety of applications [1]. A special case of the problem is that only the edges in the network have capacities (EC-network), for which extensive studies have been conducted in the past half centuries, and the best algorithms are Goldberg and Tarjan's $O(nm \log(n^2/m))$ time algorithm for real capacity [4] and Goldberg and Rao's $O(\min(n^{2/3}, m^{1/2})m \log(n^2/m) \log U_e)$ time algorithm for integral capacity [5], where n is the number of vertices, m is the number of edges, and U_e is the maximum integral capacity among the edge capacities in the network.

The problem in planar EC-networks has been addressed, and efficient algorithms have been devised by exploiting the network planarity. In particular, Hu [9] transformed the minimum (edge-)cut problem in an (s,t)-planar EC-network into the shortest path problem in the dual network of the network,

X. Hu and J. Wang (Eds.): COCOON 2008, LNCS 5092, pp. 577–586, 2008.

where an (s,t)-planar network is such a planar network that both the source s and the sink t are on the same face. Hassin [7] observed that Hu's algorithm actually computes the maximum flow. Klein *et al* [11] presented a linear algorithm for the single shortest path problem in planar networks. This leads to an $O(n)$ time algorithm for the problem in (s,t)-planar EC-networks [11]. It is not difficult to see that there is an $O(n \log n)$ algorithm for the problem in a general undirected planar EC-network by incorporating the results due to Hassin and Johnson [8] and Klein *et al* [11]. Borradaile and Klein [3] provided an $O(n \log n)$ time algorithm for the problem in a general directed planar EC-network.

It is well known that the maximum flow problem in a general EVC-network can be easily reduced to the maximum flow problem in another EC-network [1]. However, this reduction does not maintain the network planarity if it is applied to a planar network [2]. As a result, the network's planarity can not be exploited if the traditional reduction is applied, and the maximum flow problem in a planar EVC-network takes $O(n^2 \log n)$ time or $O(n^{3/2} \log n \log(\max\{U_e, U_v\})$ time for real or integral capacity respectively, where U_v is the maximum integral capacity among the vertices in the network. Khuler and Naor [10] addressed the planarity-destruction problem of the traditional reduction. By adding some edges and vertices to the dual network of a planar EVC-network, they transformed the minimum cut problem in the primal network into the problem of finding the cut-cycle of the shortest length in the extended dual network, which can be further transformed into the shortest path problem. Their algorithm, incorporating with Klein et al 's [11] shortest path algorithm, can find the minimum cut (or the value of the maximum flow) in $O(n)$ time in an (s,t)-planar EVC-network or in $O(n \log n)$ time in a general planar EVC-network. For an (s,t)-planar EVC-network, they also proposed an algorithm for computing the maximum flow, which can be implemented in $O(n \log \log n)$ time by making use of Han's sorting algorithm [6]. Inspired by Khuler and Naor's transformation, Zhang et al [14] proposed a maintaining-planarity reduction that reduces the minimum cut problem in an undirected planar EVC-network to the minimum edge-cut problem in another planar EC-network. However, finding an algorithm for computing the maximum flow in a general planar EVC-network that takes the advantage of the network's planarity was open until a solution in this paper is proposed.

In this paper we show that the maximum flow in the primal planar EVC-network can be computed by finding a maximum flow in the auxiliary planar EC-network introduced in [14], mapping the flow to a pseudo-flow in the primal EVC-network, and then canceling cycle-flows in the pseudo-flow. Thus, the maximum flow in a general undirected planar EVC-network can be found in $O(n \log n)$ time. We then provide an $O(n)$ time algorithm for canceling cycle-flows in (s,t)-planar networks by showing that the maximum flow in an undirected (s,t)-planar EVC-network can be found in $O(n)$ time. Consequently, the maximum flow problems in undirected planar EVC-networks are as easy as the problems in undirected planar EC-networks in terms of computational complexity.

The rest of the paper is organized as follows. In Section 2 we introduce necessary notations and notions. In Section 3 we show how to compute the maximum flow in an undirected planar EVC-network. In Section 4 we propose an $O(n)$ time algorithm for the problem in an undirected (s,t)-planar EVC-network. In Section 5 we conclude our discussions.

2 Preliminaries

In this section we introduce some notations and concepts which are necessary for the rest of discussions.

Definition 1. *An undirected graph $G = (V, E)$ consists of a set V of vertices and a set of edges E whose elements are pairs of distinct vertices. Denote n and m the number of vertices and edges in N. A path in a graph G is a subgraph of G consisting of a sequence of vertices $v_1 - v_2 - \cdots - v_r$ and the edges between each consecutive pair of vertices in the sequence. A simple path is a path without any repetition of vertices. A cycle is a simple path $v_1 - v_2 - \cdots - v_r$ together with edge (v_1, v_r).*

Definition 2. *An EVC-network $N = (G, c, u, s, t)$ is an undirected graph $G = (V, E)$ with two specified vertices, the source vertex s and the sink vertex t, respectively, an edge capacity function $c : E \mapsto R^+ \cup \{0\}$, an vertex capacity function $u : V \mapsto R^+$ such that $u(s) = u(t) = \infty$. An AC-network with edge capacity only is a special one in which the capacities of all vertices are ∞.*

Definition 3. *A pseudo-flow f in a network N is a function $f : V \times V \to R$ such that:*

$$f(i,j) = f(j,i) = 0, (i,j) \notin E \tag{1}$$

$$0 \le f(i,j) + f(j,i) \le c(i,j) \qquad \forall (i,j) \in E \tag{2}$$

$$\sum_{k:(k,i)\in E} f(k,i) = \sum_{j:(i,j)\in E} f(i,j) \qquad \forall i \in V \setminus \{s,t\} \tag{3}$$

Formulas (1) and (2) are referred to as the edge-capacity constraint, while Formula (3) is referred to as the flow-conservation constraint. The value of a pseudo-flow f is defined as the net flow f into t, i.e., $val(f) = \sum_{i:(i,t)\in E} f(i,t) - \sum_{j:(t,j)\in E} f(t,j)$. A path from s to t is referred to as an augmenting path if the pseudo-flow in each edge of this path is non-zero.

Definition 4. *A pseudo-flow f in a flow network N is a flow if it meets:*

$$\sum_{j:(i,j)\in E} f(i,j) \le u(i) \qquad \forall i \in V - \{s,t\} \tag{4}$$

Formula (4) is the vertex-capacity constraint of the flow. A flow f is the maximum flow if its value $val(f)$ is maximized.

Definition 5. *A path flow f^p in a flow network N on the set $\{P\}$ of all directed paths from s to t is defined as $f^p : \{P\} \to R^+ \cup \{0\}$. Specifically, a flow on a cycle of a non-simple path is called a cycle-flow. A directed path P from s to t such that $f^p(P) \geq 0$ is called an augmenting path.*

Theorem 1 ([1]). *Flow decomposition theorem: Given a flow network N, every path flow function has a unique representation as a pseudo-flow (defined on the edge set). Conversely, every pseudo-flow function in a network can be represented by at most $n+m$ path-flows, and among them, there are at most m cycle-flows.*

Definition 6. *Canceling cycle-flows within a pseudo-flow f in a network N is to decrease the values of f on edges along some cycles such that there is no cycle-flows in its path-flow representation. If the path flow representation of a pseudo-flow f contains no cycle-flows, f is called an acyclic pseudo-flow.*

Lemma 1 ([1]). *A pseudo-flow is an acyclic pseudo-flow if all its augmenting paths are simple.*

Definition 7. *Given a flow network N, a cut C in N is a minimal collection of vertices and edges whose deletion separates s from t in the resulting network. A cut consisting of only edges is an edge-cut. The sum of the capacities of the terms in a cut C is called the capacity of the cut. A minimum cut is a cut of the minimum capacity among all cuts. A minimum edge-cut is an edge-cut of the minimum capacity among all the edge-cuts.*

Definition 8. *A network is said to be planar if it can be embedded on a plane such that no two edges cross with each other. A planar network partitions the plane into a number of connected regions, and each of this regions is referred to as a face. We say the border $B(F)$ of a face F the set of edges that separate F from other parts of the plane. Two faces are neighbors if their borders share some common edges.*

3 Computing the Maximum Flow in an Undirected Planar EVC-Network

In this section we propose an algorithm for computing the maximum flow in an undirected planar EVC-network N with the aid of an auxiliary planar EC-network. The auxiliary planar EC-network, denoted as N_e, is constructed as follows. (1) Given a vertex $v \in V - \{s,t\}$ of degree d in N, replace v with a cycle consisting of d vertices $v_1, v_2, \cdots, v_d$ and d edges $(v_i, v_{i+1 \mod d}), 1 \leq i \leq d$. The edge capacity of each of these edges is $u(v)/2$. (2) The edges in N incident to v are now linked to the vertices of the cycle $v_1, v_2, \cdots, v_d$ one by one in the same clockwise order as they link to v (Fig. 1). Note that if $d = 2$, the "cycle" formed by $v_1 - v_2 - v_1$ degenerates to an edge and its capacity is $u(v)$.

Definition 9. *N_e is called the extended network of N, and the cycle in N_e formed by $v_1, v_2 \cdots, v_d$ is called the corresponding chain-cycle of $v \in N$.*

It is trivial to verify that N_e can be constructed from N in $O(n)$ time.

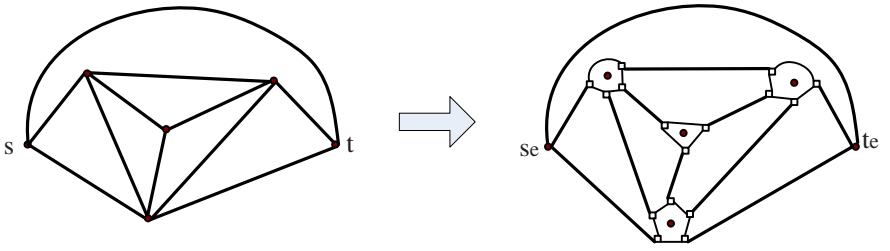

Fig. 1. The construction of an extended network

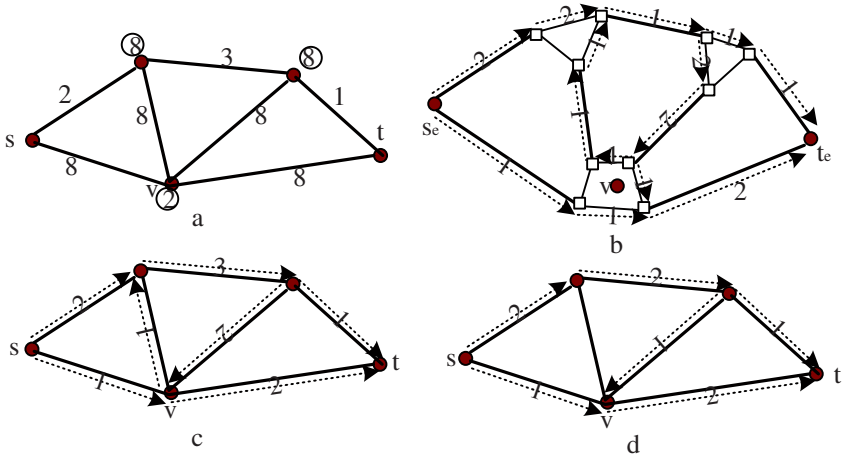

Fig. 2. (a) The original network N, (b) the extended network N_e with a flow, (c) a pseudo-flow in N is obtained from the flow in N_e, (d) a flow is derived by canceling the cycle-flow

Theorem 2. *[14] The capacity of the minimum cut N is equal to the capacity of the minimum edge-cut problem in N_e, i.e., the value of the maximum flow in N is equal to the value of the maximum flow in N_e.*

Our algorithm is based on the above theorem. Let us consider a flow in N_e. If there is a flow f_e in N_e, there is a corresponding function $f \in N, \forall e \in N, f(e) = f_e(e_e)$, where e_e is the corresponding edge of e in N_e. It is not difficult to verify that f follows the edge-capacity and flow-conversation constraints in N, thus, it is a pseudo-flow. However, f may violate the vertex-capacity constraint in N. Fig. 2 illustrates such a violation. Fig. 2(a) is a primal network N, where the value on each edge is its capacity, and the cycled value at each vertex is the capacity of the vertex. Fig. 2(b) is the extended network N_e of N and there is a maximum flow on it. Consider vertex v in N or its corresponding chain-cycle in N_e. The amount of incoming flow to v is 3, exceeding its capacity of 2. Thus, a pseudo-flow f directly obtained from a flow f_e in N_e is not a flow in N. In what follows we consider how to derive a flow from f.

Theorem 3. *If f is an acyclic pseudo-flow, then f follows vertex capacity constraint in N and is a flow.*

Proof. Without loss of generality, consider a chain-cycle in N_e, which corresponds a vertex v in N. Since s and t do not fall inside a chain-cycle, we can draw a line that cross it and separates s from t in the plane. When (a part of) flow f_e goes into the chain-cycle and then out from it, two cases may arise.

Case 1. If the flow f_e goes through the chain-cycle from one side of the plane to the other side (Fig. 3), the amount of the flow through the chain-cycle cannot be greater than the sum of the capacities of two bottleneck edges in the chain-cycle (two thick edges in Fig. 3) due to the edge-capacity constraint. Since the capacity of each edge in the chain-cycle is a half of the capacity of the corresponding vertex v in N. If the chain-cycle is treated as v, the corresponding pseudo-flow f satisfies the vertex-capacity constraint at v.

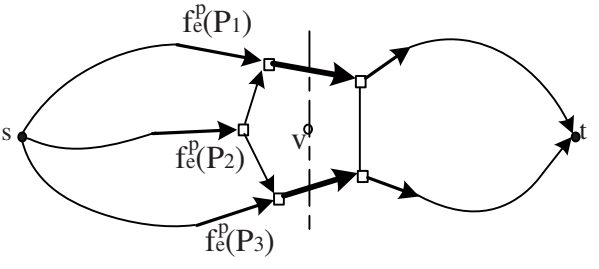

Fig. 3. Case 1. Flow runs through a chain-cycle from one side of the plane to the other side

Case 2. Opposite to case 1, without loss of generality, assume that on the left half plane, there are two path-flows $f_e^p(P_1)$ and $f_e^p(P_2)$ entering the chain-cycle and one path-flow $f_e^p(P_3)$ outgoing from the chain-cycle between the two entering path-flows(Fig. 4). The path-flows $f_e^p(P_1)$ and $f_e^p(P_2)$ and the border of the chain-cycle form a closed region. The path-flow $f_e^p(P_3)$ must go from inside the region to outside of the region because it eventually will reach the sink. Thus, if the chain-cycle is treated as the corresponding vertex v in N, a cycle-flow must be formed in the corresponding pseudo-flow f(Fig. 4).

The above observations indicate that, if f is an acyclic pseudo-flow, case 2 would not arise, and the vertex capacity constraint will not be violated.

If f is not an acyclic pseudo-flow, we can cancel its cycle-flows to make it acyclic. Thus a flow in N can be obtained as follows. First, compute a pseudo-flow f from a flow f_e in N_e, followed by cancelling the cycle-flows of f to get an acyclic pseudo-flow f_a. It is easy to verify that f_a is a flow in N. Fig. 2 depicts this procedure. In Fig. 2(c), a pseudo-flow f in N is obtained from the flow in N_e. Case 2 arises at vertex v and there is a cycle-flow. In Fig. 2(d), the cycle-flow is canceled and a flow f_a in N is obtained.

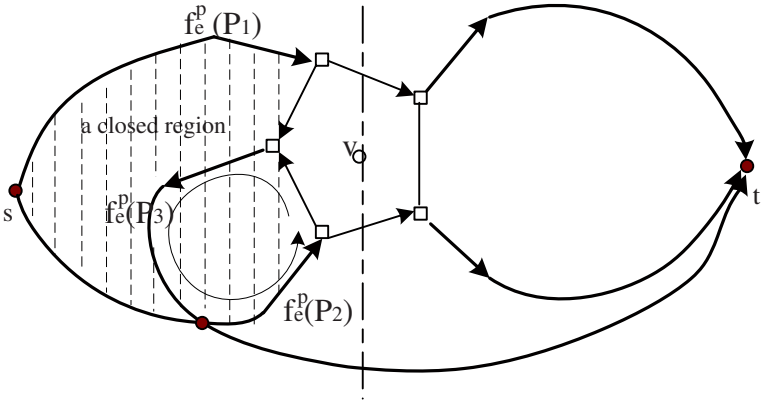

Fig. 4. Case 2. Outgoing path-flows from the chain-cycle exist between path-flows entering into the chain-cycle

Note that canceling cycle-flows does not change the value of a pseudo-flow. So if f_e is a maximum flow in N_e, then f_a is a maximum flow in N.

The algorithm is described as follows.

Algorithm Edge_Vertex_Capacitied_Max_Flow(N)
Begin
1. Construct the extended network N_e;
2. Compute a maximum flow f_e in N_e;
3. Map f_e to the edges in N to get a pseudo-flow f;
4. Cancel cycle flows in f to get a maximum flow f_a in N;
End.

Lemma 2. *[13] A pseudo-flow can be converted into an acyclic pseudo-flow of the same value in $O(m \log n)$ time.*

Theorem 4. *The maximum flow in an undirected planar EVC-network can be computed in $O(n \log n)$ time.*

4 An $O(n)$ Algorithm for Undirected (s,t)-Planar EVC-Networks

In this section, we aim to devise an efficient algorithm by taking advantage of the special properties of (s,t)-planar networks. For an (s,t)-planar network, we assume that s and t are laid in the outer face, a new edge (s,t) is introduced with $c(s,t) = 0$, the new finite face is denoted by s', and the outer face is denoted by t'.

Definition 10. *The dual network $N' = (G'(V', E'), l)$ of a (s,t)-planar network N is a planar network defined as follows. The dual vertex set V' is the set of*

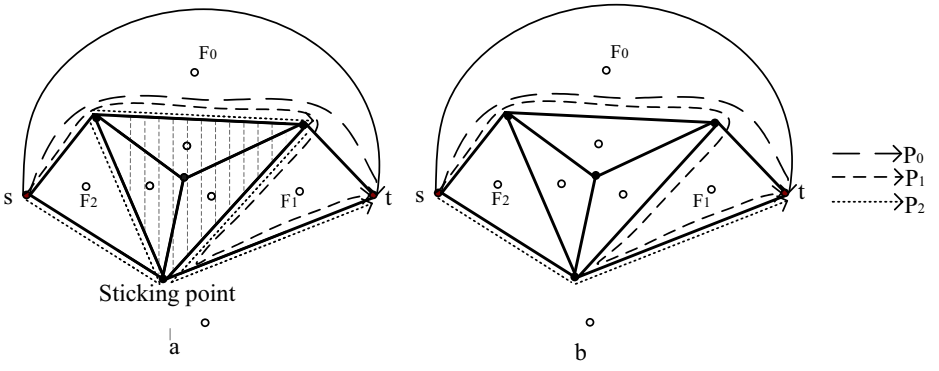

Fig. 5. Check non-simple augmenting path by visiting faces one by one: (a) A non-simple augmenting path P_2 is found, (b) P_2 is made simple by removing its traps

faces in N. For each edge $(i,j) \in E$, let i' be the face to the left of (i,j) when walking from i to j and j' be the face to the right, (i',j') is the dual edge of (i,j) in N', and its length is $l(i',j') = c(i,j)$. For each dual vertex i' in N', the length of the shortest path from s' to i' is called distance of i', denoted as $d(i')$.

Given an undirected (s,t)-planar network N and its dual network N', a label function in N' $h : V' \mapsto R$ with arbitrary value on each vertex has the following property.

Lemma 3. *[7] If two functions $\lambda : V \times V \to R$ and $f : V \times V \to R$ are defined as: $\lambda(i,j) = h(j') - h(i')$, $\lambda(t,s) = h(t') - h(s')$; $f(i,j) = \max\{\lambda(i,j), 0\}$, $f(t,s) = \max\{\lambda(t,s), 0\}$, $f(s,t) = 0$, then f follows the flow-conservation constraint on each vertex, and the value of f is $h(t') - h(s')$. If function $h(\cdot)$ is the distance function $d(\cdot)$, and N is an EC-network, then f is a maximum flow.*

To compute the maximum flow in an undirected planar EVC-network N, we first construct the extended N_e and run a shortest path algorithm in the dual network N'_e of N_e to compute the maximum flow f_e^{max}. Meanwhile, for each dual vertex v'_e that is not a face closed by a chain-cycle in N_e, we assign it an index using the order that it is added to the shortest path tree, and give it a label with its distance in N'_e, i.e., $h(v'_e) = d(v'_e)$. We finally map the index and label of each such vertex in N'_e to its corresponding face in N. A flow f in N can be computed by using the label function $h(\cdot)$ in the way as shown in Lemma 1. It is trivial to verify that f is a pseudo-flow. In what follows we show how to find a maximum flow in N.

The proposed algorithm tranverses the faces in N in increasing order of the indices of the faces, and constructs augmenting paths using the borders of these faces. Non-simple augmenting paths are then made simple by adjusting face labels. Accordingly, the corresponding flow f is acyclic and the maximum flow is

found. We refer to this algorithm as **Make_Augmenting_Path_Simple**, which works as follows. During the process, a face is marked as *visited* if it has been visited or its label is updated. A visited face means that it will not be visited again in the future. An edge or vertex is marked as *colored* if it is in the augmenting path, which will be used to update augmenting paths and to check the non-simplicity of augmenting paths.

The algorithm visits the faces in N starting from F_0, which corresponds to the dual vertex s', a simple augmenting path $P_0 = B(F_0) - (s,t)$ is found, F_0 is marked as *visited*, all the edges and vertices in P_0 are marked as *colored*. The algorithm then visits the other faces one by one in increasing order of their indices.

Suppose that F_k has been visited with $k \geq 0$, a simple augmenting path P_j is found, and the edges and vertices in P_j are marked as *colored*, now it visits the next face F_{k+i} which is an unvisited face with the least index. Note that the neighbors of F_{k+i} with smaller indices have already been visited, so some segments of $B(F_{k+i})$ are on P_k and have been marked colored. A new augmenting path P_{j+1} is obtained by replacing the colored segments with those uncolored segments of $B(F_{k+i})$. P_{j+1} may be non-simple, which can be verified by traversing $B(F_{k+i})$ as follows. If two consecutive edges (x,y) and (y,z) on $B(F_{k+i})$ are not colored while y is colored, then P_{j+1} is not simple. Each such vertex y is marked as a *sticking point*. Each non-simple sub-path of P_{j+1} is called a *trap*, and can be easily found by a traversal along P_k that starts from each sticking point, traversing the uncolored edge incident to it first, and ends at it. After this is done, the uncolored edges and vertices in $B(F_{k+i})$ are marked *colored*. Fig. 5(a) gives an illustration of the process, in which when face F_2 is visited, a trap is found.

To make augmenting path P_{j+1} simple, all the labels of faces enclosed by the trap are set to be $h(F_{k+i})$, and they are then marked as *visited* so that they will be not visited again. This label modification results in that the shared edges between $B(F_{k+i})$ and the trap have zero-flows. In the end, the traps in P_{k+1} are removed from it and a simple augmenting path P^*_{k+1} is obtained (Fig. 5(b)).

Lemma 4. *Algorithm Make_Augmenting_Path_Simple runs in $O(n)$ time.*

The proof of Lemma 4 is omitted due to the space limitation. In summary, we have the following theorem.

Theorem 5. *Given an undirected (s,t)-planar ENC-network N, there is an optimal algorithm for computing the maximum flow, which takes $O(n)$ time.*

5 Concluding Remarks

In this paper we considered the general version of the maximum flow problem in an undirected planar EVC-network, and proposed an $O(n \log n)$ time algorithm for the problem. It is still open to whether there is any efficient algorithm for the problems in directed planar networks running in the same amount of time.

References

1. Ahujia, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms and Applications. Prentice-Hall, New Jersey (1993)
2. Bondy, J.A., Murty, U.S.R.: Graph Theory with Applications. Elsevier, North-Holland (1976)
3. Borradaile, G., Klein, P.: An $O(n \log n)$ Algorithm for Maximum s-t Flow in a Directed Planar Graph. In: Proceedings of the 17th Annual ACM-SIAM Symopsimum on Discrete Algorithms (SODA 2006), pp. 524–533 (2006)
4. Goldberg, A.V., Tarjan, R.E.: A New Approach to the Maximum Flow Problem. Journal of the ACM 35, 921–940 (1988)
5. Goldberg, A.V., Rao, S.: Beyond the Flow Decomposition Barrier. Journal of the ACM 45, 783–797 (1998)
6. Han, Y.: Deterministic Sorting in $O(n \log \log n)$ Time and Linear Space. Journal of Algorithms 50, 96–105 (2004)
7. Hassin, R.: Maximum Flows in (s, t) Planar Networks. Information Processing Letters 13, 107 (1981)
8. Hassin, R., Johnson, D.S.: An $O(n \log^2 n)$ Algorithm for Maximum Flow in Undirected Planar Networks. SIAM Journal on Computing 14, 612–624 (1985)
9. Hu, T.C.: Integer Programing and Network Flows. Addison-Wesley, Reading (1969)
10. Khuler, S., Naor, J.: Flow in Planar Graphs with Vertex Capacities. Algoirthmica 11, 200–225 (1994)
11. Klein, P., Rao, S.B., Rauch-Henzinger, M., Subramanian, S.: Faster Shortest-Path Algorithms for Planar Graphs. Journal of Computer and System Science 55, 3–23 (1997)
12. Reif, J.H.: Minimum $s-t$ Cut of a Planar Undirected Network in $O(n \log^2 n)$ Time. SIAM Journal on Computing 12, 71–81 (1983)
13. Sleator, D.D., Tarjan, R.E.: A Data Structure for Dynamic Tree. Journal of Computer and System Science 26, 362–391 (1983)
14. Zhang, X., Liang, W., Jiang, H.: Flow Equivalent Trees in Node-Edge-Capacitied Undirected Planar Graphs. Information Processing Letters 100, 100–115 (2006)