Flow in Planar Graphs with Vertex Capacities

Samir Khuller and Joseph (Seffi) Naor

Abstract. Max-flow in planar graphs has always been studied with the assumption that there are capacities only on the edges. Here we consider a more general version of the problem when the vertices as well as edges have capacity constraints. In the context of general graphs considering only edge capacities is not restrictive, since the vertex-capacity problem can be reduced to the edge-capacity problem. However, in the case of planar graphs this reduction does not maintain planarity and cannot be used. We study different versions of the planar flow problem (all of which have been extensively investigated in the context of edge capacities).


1. Introduction. The computation of a maximum flow in a graph has been an important and well-studied problem, both in the fields of computer science and operations research. Many efficient algorithms have been developed to solve this problem, see, e.g., [GTT]. In this paper we concentrate on flow in planar graphs. Research on planar flow is motivated by the fact that more efficient algorithms, both sequential and parallel, can be developed by exploiting the planar structure of the graph. This is important, in particular for parallel algorithms, since maximum flow in general graphs was shown to be P-complete [GSS]. The planar flow algorithms are not only “good” because they are extremely efficient, but they are also very elegant. Planar networks also arise in practical contexts such as VLSI design and communication networks, and hence it is of interest to find fast flow algorithms for this class of graphs.

In the popular formulation of the planar flow problem a single source vertex $s$ and a sink $t$ are considered. Each edge has a capacity, and one wishes to find the max-flow from $s$ to $t$. This problem has been extensively investigated by many researchers starting from the work by Ford and Fulkerson [FF] who developed an $O(n^2)$-time algorithm for the special case of $st$-graphs (when the source and sink are on the same face). This algorithm was later improved to $O(n \log n)$ time in [IS]. By introducing the concept of potentials, Hassin [H] gave an elegant algorithm for the general case.

Received August 3, 1990; revised June 30, 1991. Communicated by Harold N. Gabow.
algorithm that runs in $O(n \sqrt{\log n})$ time using Frederickson's shortest-path algorithm [F]. Itai and Shiloach [IS] also developed an algorithm to find a max-flow in an undirected planar graph when the source and sink are not on the same face. This algorithm was improved by Reif [R] who gave an algorithm to find the value of the max-flow in $O(n \log^2 n)$ time. Hassin and Johnson [HJ] completed the picture by giving an $O(n \log^2 n)$ algorithm to compute the flow function as well. Frederickson speeded up both these algorithms by an $O(\log n)$ factor, by giving faster shortest-path algorithms [F]. The problem of finding a minimum cut in a directed planar graph turned out to be much harder and was first solved by Johnson [J] (both sequentially and in NC) who also gave algorithms to compute the flow function. Recently, Miller and Naor [MN] have pointed out that the general maximum-flow problem in planar graphs is when there are many sources and sinks. They showed that when demands are fixed, the problem can be reduced to a "circulation problem" (with lower bounds on edge capacities), and also gave an efficient algorithm for this case. Note that the multiple source–sink problem cannot be reduced to the single source–sink version since the reduction may destroy planarity.

In this paper we consider the version of the problem in which the vertices as well as edges have capacity constraints. Vertex capacities may arise in various contexts such as computing vertex disjoint paths in graphs [KS], and in various network situations when the vertices denote switches and have an upper bound on their capacities. For the case of general graphs this problem can be reduced to the version with only edges having capacity constraints by a simple idea of "splitting" vertices into two and forcing all the flow to pass through a "bottleneck" edge in between. In planar graphs this reduction may destroy the planarity of the graph and thus cannot be used. (The reduction is described on p. 205 of [BM] from which the violation of planarity is obvious.)

We show how to exploit the structure of the planar graph to develop efficient algorithms for the problem.

Notice that in the case of general graphs, as opposed to planar graphs, the single source–sink problem with edge capacities is usually the "basic" problem because most other formulations of the flow problem can be easily reduced to this problem. An example where this is not true is the problem of finding the maximum flow between each pair of vertices. The famous result of Gomory and Hu [GH1] does not hold for graphs with vertex capacities because the reduction from vertex capacities to edge capacities results in edges that are not symmetric. However, Granot and Hassin [GH2] showed how to extend the Gomory–Hu cut tree to graphs with vertex capacities.

An application where vertex capacities play an important role is in reconfiguring VLSI/WSI (Wafer Scale Integration) arrays. Assume that the processors on a wafer are configured in the form of a grid, and, due to yield problems, some are going to be faulty. Instead of treating the whole wafer as defective, the nonfaulty processors can be reconfigured in the form of a grid. We assume that multiple data tracks are allowed along every grid line. It was shown in [RBK] that, in this context, the reconfiguration problem can be abstracted combinatorially as finding a set of vertex disjoint paths from the faulty processors (the sources) to the