# A Uniform Framework for Approximating Weighted Connectivity Problems

Samir Khuller * Balaji Raghavachari † An Zhu ‡

**Abstract**

We introduce a new algorithmic technique that applies to several graph connectivity problems. Its power is demonstrated by experimental studies of the minimum-weight strongly-connected spanning subgraph problem and the minimum-weight augmentation problem. Even though we are unable to improve the approximation ratios for these problems, our studies indicate that the new method generates significantly better solutions than the current known approximation algorithms, and yields solutions very close to optimal. We believe that our technique will eventually lead to algorithms that improve the performance ratios as well.

## 1 Introduction

Let a weighted graph $G = (V, E)$ represent all the feasible links of a potential communications network. A minimum spanning tree in $G$ is the cheapest connected subgraph, i.e., the cheapest network that will allow the sites to communicate. Such a network is highly susceptible to failures, since it cannot even survive a single link or site failure. For more reliable communication, one desires spanning subgraphs of higher connectivities. A network of edge-connectivity (vertex-connectivity) $k$ continues to allow communication between functioning sites even after as many as $k-1$ links (sites) have failed. Given an arbitrary graph $G$ with non-negative edge-weights, and an integer $k$, we consider the problem of finding a minimum-weight $k$-connected spanning subgraph. Due to lack of space, we have not cited the papers in this area, and the reader is referred to a survey [3] for relevant papers.

In directed graphs, the corresponding problem is that of finding a minimum weight strongly connected spanning subgraph (SCSS) and its generalizations to higher connectivities. A strongly connected graph is one which has a directed path, between every ordered pair of vertices. Due to lack of space in this abstract, we will only discuss our method as applied to the strongly connected spanning subgraph problem. Since this problem is $NP$-hard, we will discuss polynomial time heuristics. One heuristic for this problem works as follows: fix a root $r$ and find a minimum weight in-branching $B_I$ and a minimum weight out-branching $B_O$, and take the union of the two branchings. Frederickson and JáJá [1] used this idea to obtain a 2-approximation for the SCSS problem. Observe that the weight of each branching is at most the weight of the minimum strongly connected subgraph; thus we get an approximation factor of 2. A lot of work has been done on the design of approximation algorithms for undirected graphs, but very little is known about directed graphs.

## 2 The strong connectivity algorithm

The input is a directed graph $G = (V, E)$ with weights on the edges. The goal is to find an SCSS of $G$ of minimum weight.

Let $r$ be an arbitrary root vertex of $G$. Let $B_I$ denote a minimum weight in-branching in $G$, rooted at $r$. Let $w(B_I)$ denote the weight of $B_I$. Similarly, $B_O$ denotes a minimum weight out-branching rooted at $r$ and $w(B_O)$ denotes its weight. Let $s = w(B_I) + w(B_O)$ be the sum of the weights of the branchings.

**Definition:** Let $e \in E$ be an edge with $w(e) > 0$. Let $B'_I$ and $B'_O$ be the minimum weight in-branching and out-branching's in $G$ if $w(e)$ is *reduced* to 0. Define **drop(e)** as follows:

$$\text{drop(e)} = \frac{(w(B_I) + w(B_O)) - (w(B'_I) + w(B'_O))}{w(e)}$$

In other words, *drop(e)* is defined as the ratio of the *decrease* in the sum of the weights of the two branchings when $w(e)$ is made zero, to $w(e)$. Intuitively, the higher the value of *drop(e)*, the more it is "shared" by the two branchings. Clearly, *drop(e)* is at most 2 because, in the extreme case, $e$ belongs to both $B_I$ and $B_O$, and reducing its weight to zero decreases the total weight of these two branchings by $2w(e)$). Also, *drop($e^*$)*, the maximum of drop($e$) over all edges of non-zero weight, is at least 1.

Our algorithm starts by selecting a root vertex $r$. It then repeatedly finds an edge $e$ that maximizes drop($e$), and sets $w(e)$ to 0. This process is repeated until the

*Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. Research supported by NSF CAREER Award CCR-9501355. E-mail : samir@cs.umd.edu.

†Department of Computer Science, The University of Texas at Dallas, Richardson, TX 75083-0688. Research supported by NSF award CCR-9409625. E-mail : rbk@utdallas.edu.

‡Department of Computer Science, University of Maryland, College Park, MD 20742. Research partly supported by NSF CAREER Award CCR-9501355, and a Senior Summer Scholarship from University of Maryland. E-mail : anzhu@wam.umd.edu.

weight of both $B_I$ and $B_O$ become 0. The algorithm returns the union of $B_I$ and $B_O$ as its output. Currently, the best approximation ratio that we can prove is 2, the same as the old algorithm. But our experiments show that the new algorithm performs much better in practice, yielding solutions that are very close to the optimal solution.

## 3 General framework

Suppose, given an NP-hard problem $P$, we could "relax" some of the constraints from problem $P$, obtaining a set of sub-problems $P_1, P_2, \ldots, P_t$ such that, each $P_i$ can be solved optimally in polynomial time yielding a solution $S_i$. By taking the union of the solutions to the sub-problems, suppose we obtain a solution $S$ for the original problem $P$. By proving that $w(S_i) \leq k_i w(OPT)$, where $k_i$ is a constant and $OPT$ is the weight of an optimal solution to problem $P$, we obtain: $w(S) \leq \sum_{i=1}^{t} w(S_i) \leq w(OPT) \sum_{i=1}^{t} k_i$. This gives a polynomial time approximation algorithm with an approximation factor of $\sum_i k_i$. For strong connectivity, $P_1$ is a minimum-weight in-branching problem, and $P_2$ is a minimum-weight out-branching problem. In this case, $k_1 = k_2 = 1$, and the approximation ratio is 2.

Our algorithm can be applied in such a framework as follows. For each edge $e$, we measure the amount by which it affects $\sum w(S_i)$ if $w(e)$ were made 0. The ratio of the net change in $\sum w(S_i)$ to $w(e)$ is drop($e$). An edge has a high drop value if it can be utilized by several (near) optimal solutions to the subproblems. We could then choose an edge with a high drop value and include it in the solution by making its weight 0. This is repeated until there is a zero-weight solution. The algorithm then returns as a solution all the edges that were chosen in one of the steps. Problems that fit in this framework include:

**$k$-Edge connectivity:** Given a graph $G = (V, E)$ with a weight function $w$ defined on the edges, the $k$**ECSS** problem is to find a minimum weight $k$-edge-connected spanning subgraph.

**Augmentation:** Given a graph $G = (V, E)$, a subgraph $H = (V, E_H)$ of $G$, and a positive integer $k$, the $k$**AUG** problem is to find a subset of edges of $G$ with minimum total weight, whose addition to $E_H$ generates a $k$-edge-connected graph[1, 3]. When the initial subgraph $H$ has no edges then this is the $k$ECSS problem.

**Directed edge connectivity:** The input is a graph $G = (V, E)$ and a positive integer $k$. The problem is to find a min-weight subgraph which has $k$ disjoint directed paths between every ordered pair of vertices.

**Vertex connectivity:** find a minimum-weight $k$-connected spanning subgraph.

**Survivable network design:** find a minimum-weight subgraph that crosses each cut by at least a specified number of times, for which Jain [2] has given a 2-approximation algorithm. It is interesting to note that his analysis can be used to show that our algorithm also has a performance ratio of 2 for this problem.

## 4 Experimental results

We ran many experiments for strong-connectivity and augmentation problems. In the first set of experiments, we ran the algorithms on weighted graphs where the weights were chosen randomly. For each size, we ran the algorithms on 21 different randomly chosen graphs of varying densities. The second set of experiments were made on unweighted Hamiltonian graphs. We also ran similar experiments for the augmentation (2AUG) problem. The results of the experiments are shown below. "Size" denotes the number of vertices. The column marked "Old" gives the performance of the previous algorithm which tries all possible vertices as root and then selects the best solution. Column marked "D-1" gives the performance of our drop algorithm that uses an arbitrary vertex as root and the column marked "D-2" gives the performance of this algorithm when we try each vertex as root and then select the best solution. Observe that in all cases, our algorithm performs much better than the previous best algorithms.

| **SCSS problem:** % deviation from optimal | | | | | | |
|---|---|---|---|---|---|---|
| | Weighted | | | Unweighted | | |
| Size | Old | D-1 | D-2 | Old | D-1 | D-2 |
| 10 | 6.7 | 2.8 | 0.4 | 42.9 | 8.1 | 1.0 |
| 15 | 7.2 | 2.4 | 0.7 | 47.9 | 10.2 | 3.5 |
| 20 | 10.8 | 3.3 | 0.9 | 53.5 | 11.2 | 5.9 |
| 25 | 12.3 | 2.9 | 0.8 | 55.8 | 9.9 | 5.3 |

| **Augmentation:** % deviation from optimal | | | | | | |
|---|---|---|---|---|---|---|
| | Weighted | | | Unweighted | | |
| Size | Old | D-1 | D-2 | Old | D-1 | D-2 |
| 10 | 1.6 | 0.0 | 0.0 | 9.5 | 1.6 | 0.0 |
| 15 | 3.9 | 0.0 | 0.0 | 22.7 | 1.9 | 0.0 |
| 20 | 2.7 | 0.2 | 0.0 | 28.1 | 2.9 | 0.0 |
| 25 | 5.9 | 0.5 | 0.0 | 31.9 | 3.7 | 0.8 |
| 50 | 6.9 | 0.7 | 0.0 | 42.9 | 3.6 | 0.7 |

## References


[1] G.N. Frederickson and J. JáJá, "Approximation algorithms for several graph augmentation problems," *SIAM J. Comput.*, 10 (2), pp. 270–283, (1981).

[2] K. Jain, "A factor 2 approximation algorithm for the generalized Steiner network problem," FOCS 1998.

[3] S. Khuller, "Approximation algorithms for finding highly connected subgraphs," in *Approximation Algorithms for NP-hard problems*, D. Hochbaum (ed.), PWS Publishing Company (1996).