Encyclopedia of Algorithms
DOI 10.1007/978-3-642-27848-8_168-2

Gomory-Hu Trees

Debmalya Panigrahi*
Department of Computer Science, Duke University, Durham, NC, USA

Keywords Cut trees • Undirected graph connectivity • Minimum *s*-*t* cut

Years and Authors of Summarized Original Work

2007; Bhalgat, Hariharan, Kavitha, Panigrahi

Problem Definition

Let $G = (V, E)$ be an undirected graph with $|V| = n$ and $|E| = m$. The edge connectivity of two vertices $s, t \in V$, denoted by $\lambda(s, t)$, is defined as the size of the smallest cut that separates s and t; such a cut is called a minimum $s - t$ cut. Clearly, one can represent the $\lambda(s, t)$ values for all pairs of vertices s and t in a table of size $O(n^2)$. However, for reasons of efficiency, one would like to represent all the $\lambda(s, t)$ values in a more succinct manner. *Gomory-Hu trees* (also known as *cut trees*) offer one such succinct representation of linear (i.e., $O(n)$) space and constant (i.e., $O(1)$) lookup time. It has the additional advantage that apart from representing all the $\lambda(s, t)$ values, it also contains structural information from which a minimum $s - t$ cut can be retrieved easily for any pair of vertices s and t.

Formally, a Gomory-Hu tree $T = (V, F)$ of an undirected graph $G = (V, E)$ is a weighted undirected tree defined on the vertices of the graph such that the following properties are satisfied:

- For any pair of vertices $s, t \in V$, $\lambda(s, t)$ is equal to the minimum weight on an edge in the unique path connecting s to t in T. Call this edge $e(s, t)$. If there are multiple edges with the minimum weight on the s to t path in T, any one of these edges is designated as $e(s, t)$.
- For any pair of vertices s and t, the bipartition of vertices into components produced by removing $e(s, t)$ (if there are multiple candidates for $e(s, t)$, this property holds for each candidate edge) from T corresponds to a minimum $s - t$ cut in the original graph G.

To understand this definition better, consider the following example. Figure 1 shows an undirected graph and a corresponding Gomory-Hu tree. Focus on a pair of vertices, for instance, 3 and 5. Clearly, the edge (6,5) of weight 3 is a minimum-weight edge on the 3 to 5 path in the Gomory-Hu tree. It is easy to see that $\lambda(3, 5) = 3$ in the original graph. Moreover, removing edge (6,5) in the Gomory-Hu tree produces the vertex bipartition ({1,2,3,6},{4,5}), which is a cut of size 3 in the original graph.

*E-mail: debmalya@cs.duke.edu, debmalya@alum.mit.edu

DOI 10.1007/978-3-642-27848-8_168-2

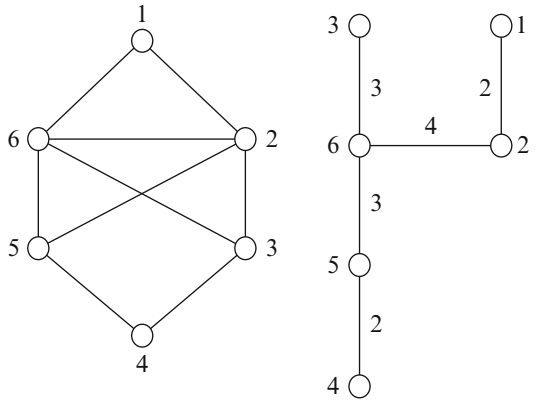

Fig. 1 An undirected graph (*left*) and a corresponding Gomory-Hu tree (*right*)

It is not immediate that such Gomory-Hu trees exist for all undirected graphs. In a classical result in 1961, Gomory and Hu [8] showed that not only do such trees exist for all undirected graphs but that they can also be computed using $n-1$ minimum s-t cut (or equivalently maximum s-t flow) computations. In fact, a graph can have multiple Gomory-Hu trees.

All previous algorithms for constructing Gomory-Hu trees for undirected graphs used maximum-flow subroutines. Gomory and Hu gave an algorithm to compute a cut tree T using $n-1$ maximum-flow computations and graph contractions. Gusfield [9] proposed an algorithm that does not use graph contractions; all $n-1$ maximum-flow computations are performed on the input graph. Goldberg and Tsioutsiouliklis [7] did an experimental study of the algorithms due to Gomory and Hu and due to Gusfield for the cut tree problem and described efficient implementations of these algorithms. Examples were shown by Benczúr [1] that cut trees do not exist for directed graphs.

Any maximum-flow-based approach for constructing a Gomory-Hu tree would have a running time of $(n-1)$ times the time for computing a single maximum flow. Till now, faster algorithms for Gomory-Hu trees were by-products of faster algorithms for computing a maximum flow. The current fastest $\tilde{O}(m + n\lambda(s,t))$ (polylog n factors ignored in $\tilde{O}$ notation) maximum-flow algorithm, due to Karger and Levine [11], yields the current best expected running time of $\tilde{O}(n^3)$ for Gomory-Hu tree construction on simple unweighted graphs with n vertices. Bhalgat et al. [2] improved this time complexity to $\tilde{O}(mn)$. Note that both Karger and Levine's algorithm and Bhalgat et al.'s algorithm are randomized Las Vegas algorithms. The fastest deterministic algorithm for the Gomory-Hu tree construction problem is a by-product of Goldberg and Rao's maximum-flow algorithm [6] and has a running time of $\tilde{O}(nm^{1/2}\min(m, n^{3/2}))$.

Since the publication of the results of Bhalgat et al. [2], it has been observed that the maximum-flow subroutine of Karger and Levine [11] can also be used to obtain an $\tilde{O}(mn)$ time Las Vegas algorithm for constructing the Gomory-Hu tree of an unweighted graph. However, this algorithm does not yield partial Gomory-Hu trees which are defined below. For planar undirected graphs, Borradaile et al. [3] gave an $\tilde{O}(mn)$ time algorithm for constructing a Gomory-Hu tree.

It is important to note that in spite of the tremendous recent progress in approximate maximum s-t flow (or approximate minimum s-t cut) computation, this does not immediately translate to an improved algorithm for approximate Gomory-Hu tree construction. This is because of two reasons: first, the property of uncrossability of minimum s-t cuts used by Gomory and Hu in their minimum s-t cut based cut tree construction algorithm does not hold for approximate minimum s-t cuts, and

DOI 10.1007/978-3-642-27848-8_168-2

second, the errors introduced in individual minimum *s*-*t* cut computation can add up to create large errors in the Gomory-Hu tree.

Key Results

Bhalgat et al. [2] considered the problem of designing an efficient algorithm for constructing a Gomory-Hu tree on unweighted undirected graphs. The main theorem shown in this entry is the following.

Theorem 1. *Let $G = (V, E)$ be a simple unweighted graph with m edges and n vertices. Then a Gomory-Hu tree for G can be built in expected time $\tilde{O}(mn)$.*

Their algorithm is always faster by a factor of $\tilde{\Omega}(n^{2/9})$ (polylog n factors ignored in $\tilde{\Omega}$ notation) compared to the previous best algorithm.

Instead of using maximum-flow subroutines, they use a Steiner connectivity algorithm. The *Steiner connectivity* of a set of vertices S (called the *Steiner set*) in an undirected graph is the minimum size of a cut which splits S into two parts; such a cut is called a *minimum Steiner cut*. Generalizing a tree-packing algorithm given by Gabow [5] for finding the edge connectivity of a graph, Cole and Hariharan [4] gave an algorithm for finding the Steiner connectivity k of a set of vertices in either undirected or directed Eulerian unweighted graphs in $\tilde{O}(mk^2)$ time. (For undirected graphs, their algorithm runs a little faster in time $\tilde{O}(m + nk^3)$.) Bhalgat et al. improved this result and gave the following theorem.

Theorem 2. *In an undirected or directed Eulerian unweighted graph, the Steiner connectivity k of a set of vertices can be determined in time $\tilde{O}(mk)$.*

The algorithm in [4] was used by Hariharan et al. [10] to design an algorithm with expected running time $\tilde{O}(m + nk^3)$ to compute a *partial* Gomory-Hu tree for representing the $\lambda(s, t)$ values for all pairs of vertices s, t that satisfied $\lambda(s, t) \leq k$. Replacing the algorithm in [4] by the new algorithm for computing Steiner connectivity yields an algorithm to compute a partial Gomory-Hu tree in expected running time $\tilde{O}(m + nk^2)$. Bhalgat et al. showed that using a more detailed analysis, this result can be improved to give the following theorem.

Theorem 3. *The partial Gomory-Hu tree of an undirected unweighted graph to represent all $\lambda(s, t)$ values not exceeding k can be constructed in expected time $\tilde{O}(mk)$.*

Since $\lambda(s, t) < n$ for all s, t vertex pairs in an unweighted (and simple) graph, setting k to n in Theorem 3 implies Theorem 1.

Applications

Gomory-Hu trees have many applications in multiterminal network flows and are an important data structure in graph connectivity literature.

DOI 10.1007/978-3-642-27848-8_168-2

Open Problems

The problem of derandomizing the algorithm due to Bhalgat et al. [2] to produce an $\tilde{O}(mn)$ time deterministic algorithm for constructing Gomory-Hu trees for unweighted undirected graphs remains open. The other main challenge is to extend the results in [2] to weighted graphs.

Experimental Results

Goldberg and Tsioutsiouliklis [7] did an extensive experimental study of the cut tree algorithms due to Gomory and Hu [8] and that due to Gusfield [9]. They showed how to efficiently implement these algorithms and also introduced and evaluated heuristics for speeding up the algorithms. Their general observation was that while Gusfield's algorithm is faster in many situations, Gomory and Hu's algorithm is more robust. For more detailed results of their experiments, refer to [7].

No experimental results are reported for the algorithm due to Bhalgat et al. [2].

Recommended Reading

1. Benczúr AA (1995) Counterexamples for directed and node capacitated cut-trees. SIAM J Comput 24(3):505–510
2. Bhalgat A, Hariharan R, Kavitha T, Panigrahi D (2007) An $\tilde{O}(mn)$ Gomory-Hu tree construction algorithm for unweighted graphs. In: Proceedings of the 39th annual ACM symposium on theory of computing, San Diego
3. Borradaile G, Sankowski P, Wulff-Nilsen C (2010) Min st-cut Oracle for planar graphs with near-linear preprocessing time. In: Proceedings of the 51th annual IEEE symposium on foundations of computer science, Las Vegas, pp 601–610
4. Cole R, Hariharan R (2003) A fast algorithm for computing steiner edge connectivity. In: Proceedings of the 35th annual ACM symposium on theory of computing, San Diego, pp 167–176
5. Gabow HN (1995) A matroid approach to finding edge connectivity and packing arborescences. J Comput Syst Sci 50:259–273
6. Goldberg AV, Rao S (1998) Beyond the flow decomposition barrier. J ACM 45(5):783–797
7. Goldberg AV, Tsioutsiouliklis K (2001) Cut tree algorithms: an experimental study. J Algorithms 38(1):51–83
8. Gomory RE, Hu TC (1961) Multi-terminal network flows. J Soc Ind Appl Math 9(4):551–570
9. Gusfield D (1990) Very simple methods for all pairs network flow analysis. SIAM J Comput 19(1):143–155
10. Hariharan R, Kavitha T, Panigrahi D (2007) Efficient algorithms for computing all low *s-t* edge connectivities and related problems. In: Proceedings of the 18th annual ACM-SIAM symposium on discrete algorithms, New Orleans, pp 127–136
11. Karger D, Levine M (2002) Random sampling in residual graphs. In: Proceedings of the 34th annual ACM symposium on theory of computing, Montreal, pp 63–66