Separators (1988, 1999; Leighton, Rao)

Goran Konjevod, Arizona State University, thrackle.eas.asu.edu

entry editor: Chandra Chekuri

INDEX TERMS: separators, balanced cuts, multicommodity max-flow min-cut theorems, sparsest cut, graph partitioning, minimum cut linear arrangement, bisection, feedback arc set, elimination orderings, approximation algorithms, ℓ_1-embeddings

SYNONYMS: balanced cuts

1 PROBLEM DEFINITION

The (balanced) separator problem asks for a cut of minimum (edge)-weight in a graph, such that the two shores of the cut have approximately equal (node)-weight.

Formally, given an undirected graph $G = (V, E)$, with a nonnegative edge-weight function $c : E \to \mathbb{R}_+$, a nonnegative node-weight function $\pi : V \to \mathbb{R}_+$, and a constant $b \leq 1/2$, we say a cut $(S : V \setminus S)$ is *b-balanced*, or a $(b, 1-b)$-*separator*, if $b\pi(V) \leq \pi(S) \leq (1-b)\pi(V)$ (where we write $\pi(S)$ for $\sum_{v \in S} \pi(v)$).

Problem 1 (b-balanced separator)**.** INPUT*: Edge- and node-weighted graph $G = (V, E, c, \pi)$, constant $b \leq 1/2$.*
OUTPUT*: A b-balanced cut $(S : V \setminus S)$. Goal: minimize the edge weight $c(\delta(S))$.*

Closely related is the *product sparsest cut problem.*

Problem 2 ((Product) Sparsest cut)**.** INPUT*: Edge- and node-weighted graph $G = (V, E, c, \pi)$.*
OUTPUT*: A cut $(S : V \setminus S)$ minimizing the ratio-cost $\frac{c(\delta(S))}{\pi(S)\pi(V \setminus S)}$.*

Problem 2 is the most general version of sparsest cut solved by Leighton and Rao. Setting all node weights are equal to 1 leads to the uniform version, Problem 3.

Problem 3 ((Uniform) Sparsest cut)**.** INPUT*: Edge-weighted graph $G = (V, E, c)$.*
OUTPUT*: A cut $(S : V \setminus S)$ minimizing the ratio-cost $\frac{c(\delta(S))}{|S||V \setminus S|}$.*

Sparsest cut arises as the (integral version of the) linear programming dual of *concurrent multicommodity flow* (Problem 4). An instance of a multicommodity flow problem is defined on an edge-weighted graph by specifying for each of k *commodities* a *source* $s_i \in V$, a *sink* $t_i \in V$, and a *demand* D_i. A feasible solution to the multicommodity flow problem defines for each commodity a flow function on E, thus routing a certain amount of flow from s_i to t_i. The edge weights represent capacities, and for each edge e, a capacity constraint is enforced: the sum of all commodities' flows through e is at most the capacity $c(e)$.

Problem 4 (Concurrent multicommodity flow)**.** INPUT*: Edge-weighted graph $G = (V, E, c)$, commodities $(s_1, t_1, D_1), \ldots (s_k, t_k, D_k)$.*
OUTPUT*: A multicommodity flow that routes fD_i units of commodity i from s_i to t_i for each i simultaneously, without violating the capacity of any edge. Goal: maximize f.*

Problem 4 can be solved in polynomial time by linear programming, and approximated arbitrarily well by several more efficient combinatorial algorithms (Section 2.2). The maximum value f for which there exists a multicommodity flow is called the *max-flow* of the instance. The *min-cut* is the minimum ratio $\frac{c(\delta(S))}{D(S,V\setminus S)}$, where $D(S, V \setminus S) = \sum_{i:|\{s_i,t_i\}\cap S|=1} D_i$.

This dual interpretation motivates the most general version of the problem, the *nonuniform sparsest cut* (Problem 5).

Problem 5 ((Nonuniform) Sparsest cut)**.** INPUT*: Edge-weighted graph $G = (V, E, c)$, commodities $(s_1, t_1, D_1), \ldots (s_k, t_k, D_k)$.*
OUTPUT*: A min-cut $(S : V \setminus S)$, that is, a cut of minimum ratio-cost $\frac{c(\delta(S))}{D(S,V\setminus S)}$.*

(Most literature focuses on either the uniform or the general nonuniform version, and both of these two versions are sometimes referred to as just the "sparsest cut" problem.)

2 KEY RESULTS

Even when all (edge- and node-) weights are equal to 1, finding a minimum-weight b-balanced cut is NP-hard (for $b = 1/2$, the problem becomes *graph bisection*). Leighton and Rao [23, 24] give a pseudo-approximation algorithm for the general problem.

Theorem 1. *There is a polynomial-time algorithm that, given a weighted graph $G = (V, E, c, \pi)$, $b \le 1/2$ and $b' < \min\{b, 1/3\}$, finds a b'-balanced cut of weight $O(\frac{\log n}{b-b'})$ times the weight of the minimum b-balanced cut.*

The algorithm solves the sparsest cut problem on the given graph, puts aside the smaller-weight shore of the cut, and recurses on the larger-weight shore until both shores of the sparsest cut found have weight at most $(1-b')\pi(G)$. Now the larger-weight shore of the last iteration's sparsest cut is returned as one shore of the balanced cut, and everything else as the other shore.

Since the sparsest cut problem is itself NP-hard, Leighton and Rao first required an approximation algorithm for this problem.

Theorem 2. *There is a polynomial-time algorithm with approximation ratio $O(\log p)$ for product sparsest cut (Problem 2), where p denotes the number of nonzero-weight nodes in the graph.*

This algorithm follows immediately from Theorem 3.

Theorem 3. *There is a polynomial-time algorithm that finds a cut $(S : V \setminus S)$ with ratio-cost $\frac{c(\delta(S))}{\pi(S)\pi(V\setminus S)} \in O(f \log p)$, where f is the max-flow for the product multicommodity flow and p the number of nodes with nonzero weight.*

The proof of Theorem 3 is based on solving a linear programming formulation of the multicommodity flow problem and using the solution to construct a sparse cut.

2.1 Related results

Shahrokhi and Matula [27] gave a max-flow min-cut theorem for a special case of the multicommodity flow problem and used a similar LP-based approach to prove their result. An $O(\log n)$ upper bound for arbitrary demands was proved by Aumann and Rabani [6] and Linial et al [26]. In both cases, the solution to the dual of the multicommodity flow linear program is interpreted as a finite metric and embedded into ℓ_1 with distortion $O(\log n)$, using an embedding due to Bourgain [10]). The resulting ℓ_1 metric is a convex combination of cut metrics, from which a cut can be extracted with sparsity ratio at least as good as that of the combination.

Arora et al [5] gave an $O(\sqrt{\log n})$ pseudo-approximation algorithm for (uniform or product-weight) balanced separators, based on a semidefinite programming relaxation. For the nonuniform version, the best bound is $O(\sqrt{\log n} \log \log n)$ due to Arora et al [4]. Khot and Vishnoi [18] showed that, for the nonuniform version of the problem, the semidefinite relaxation of [5] has an integrality gap of at least $(\log \log n)^{1/6-\delta}$ for any $\delta > 0$, and further, assuming their Unique Games Conjecture, that it is NP-hard to (pseudo)-approximate the balanced separator problem to within any constant factor. The SDP integrality gap was strengthened to $\Omega(\log \log n)$ by Krauthgamer and Rabani [20]. Devanur et al [11] show an $\Omega(\log \log n)$ integrality gap for the SDP formulation even in the uniform case.

2.2 Implementation

The bottleneck in the balanced separator algorithm is solving the multicommodity flow linear program. There exists a substantial amount of work on fast approximate solutions to such linear programs [19, 22, 25]. In most of the following results, the algorithm produces a $(1+\epsilon)$-approximation, and its hidden constant depends on ϵ^{-2}. Garg and Könemann [15], Fleischer [14] and Karakostas [16] gave efficient approximation schemes for multicommodity flow and related problems, with running times $\tilde{O}((k+m)m)$ [15] and $\tilde{O}(m^2)$ [14, 16]. Benczúr and Karger [7] gave an $O(\log n)$ approximation to sparsest cut based on randomized minimum cut and running in time $\tilde{O}(n^2)$. The current fastest $O(\log n)$ sparsest cut (balanced separator) approximation is based on a primal-dual approach to semidefinite programming due to Arora and Kale [3], and runs in time $O(m+n^{3/2})$ ($\tilde{O}(m+n^{3/2})$, respectively). The same paper gives an $O(\sqrt{\log n})$ approximation in time $O(n^2)$ ($\tilde{O}(n^2)$, respectively), improving on a previous $\tilde{O}(n^2)$ algorithm of Arora et al [2]. If an $O(\log^2 n)$ approximation is sufficient, then sparsest cut can be solved in time $\tilde{O}(n^{3/2})$, and balanced separator in time $\tilde{O}(m+n^{3/2})$ [17].

3 APPLICATIONS

Many problems can be solved by using a balanced separator or sparsest cut algorithm as a subroutine. The approximation ratio of the resulting algorithm typically depends directly on the ratio of the underlying subroutine. In most cases, the graph is recursively split into pieces of balanced size. In addition to the $O(\log n)$ approximation factor required by the balanced separator algorithm, this leads to another $O(\log n)$ factor due to the recursion depth. Even et al [12] improved many results based on balanced separators by using *spreading metrics*, reducing the approximation guarantee to $O(\log n \log \log n)$ from $O(\log^2 n)$.

Some applications are listed here; where no reference is given, and for further examples, see [24].

- Minimum cut linear arrangement and minimum feedback arc set. One single algorithm provides an $O(\log^2 n)$ approximation for both of these problems.
- Minimum chordal graph completion and elimination orderings [1]. Elimination orderings are useful for solving sparse symmetric linear systems. The $O(\log^2 n)$ approximation algorithm of [1] for chordal graph completion has been improved to $O(\log n \log\log n)$ by Even et al [12].
- Balanced node cuts. The cost of a balanced cut may be measured in terms of the weight of nodes removed from the graph. The balanced separator algorithm can be easily extended to this node-weighted case.
- VLSI layout. Bhatt and Leighton [8] studied several optimization problems in VLSI layout. Recursive partitioning by a balanced separator algorithm leads to polylogarithmic approximation algorithms for crossing number, minimum layout area and other problems.
- Treewidth and pathwidth. Bodlaender et al [9] showed how to approximate treewidth within $O(\log n)$ and pathwidth within $O(\log^2 n)$ by using balanced node separators.
- Bisection. Feige and Krauthgamer [13] gave an $O(\alpha \log n)$ approximation for the minimum bisection, using any α-approximation algorithm for sparsest cut.

4 EXPERIMENTAL RESULTS

Lang and Rao [21] compared a variant of the sparsest cut algorithm from [24] to methods used in graph decomposition for VLSI design.

5 CROSS REFERENCES

None is reported. Entry editors please feel free to add some.

6 RECOMMENDED READING

Further details and pointers to additional results may be found in the survey [28].

[1] A. Agrawal, P. N. Klein, and R. Ravi. Cutting down on fill using nested dissection: provably good elimination orderings. In *Graph theory and sparse matrix computation*, IMA Volumes in mathematics and its applications, pages 31–55. Springer, 1993.

[2] S. Arora, E. Hazan, and S. Kale. Fast algorithms for approximate semide.nite programming using the multiplicative weights update method. In *FOCS '05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 339–348, Washington, DC, USA, 2005. IEEE Computer Society.

[3] S. Arora and S. Kale. A combinatorial, primal-dual approach to semidefinite programs. In *STOC '06: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, New York, NY, USA, 2007. ACM Press.

[4] S. Arora, J. R. Lee, and A. Naor. Euclidean distortion and the sparsest cut. In *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 553–562, New York, NY, USA, 2005. ACM Press.

[5] S. Arora, S. Rao, and U. Vazirani. Expander flows, geometric embeddings and graph partitioning. In *STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 222–231, New York, NY, USA, 2004. ACM Press.

[6] Y. Aumann and Y. Rabani. An (log) approximate min-cut max-flow theorem and approximation algorithm. *SIAM J. Comput.*, 27(1):291–301, 1998.

[7] A. A. Benczúr and D. R. Karger. Approximating s-t minimum cuts in $\tilde{O}$(n2) time. In *STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 47–55, New York, NY, USA, 1996. ACM Press.

[8] S. N. Bhatt and F. T. Leighton. A framework for solving vlsi graph layout problems. *J. Comput. Syst. Sci.*, 28(2):300–343, 1984.

[9] H. L. Bodlaender, J. R. Gilbert, H. Hafsteinsson, and T. Kloks. Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. *J. Algorithms*, 18(2):238–255, 1995.

[10] J. Bourgain. On Lipshitz embedding of finite metric spaces in Hilbert space. *Israel J. Math.*, 52:46–52, 1985.

[11] N. R. Devanur, S. A. Khot, R. Saket, and N. K. Vishnoi. Integrality gaps for sparsest cut and minimum linear arrangement problems. In *STOC '06: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 537–546, New York, NY, USA, 2006. ACM Press.

[12] G. Even, J. S. Naor, S. Rao, and B. Schieber. Divide-and-conquer approximation algorithms via spreading metrics. *J. ACM*, 47(4):585–616, 2000.

[13] U. Feige and R. Krauthgamer. A polylogarithmic approximation of the minimum bisection. In *FOCS '00: Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, page 105, Washington, DC, USA, 2000. IEEE Computer Society.

[14] L. Fleischer. Approximating fractional multicommodity flow independent of the number of commodities. In *FOCS '99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, page 24, Washington, DC, USA, 1999. IEEE Computer Society.

[15] N. Garg and J. Koenemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *FOCS '98: Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, page 300, Washington, DC, USA, 1998. IEEE Computer Society.

[16] G. Karakostas. Faster approximation schemes for fractional multicommodity flow problems. In *SODA '02: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 166–173, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics.

[17] R. Khandekar, S. Rao, and U. Vazirani. Graph partitioning using single commodity flows. In *STOC '06: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 385–390, New York, NY, USA, 2006. ACM Press.

[18] S. Khot and N. K. Vishnoi. The unique games conjecture, integrality gap for cut problems and embeddability of negative type metrics into l_1. In *FOCS*, pages 53–62, 2005.

[19] P. N. Klein, S. A. Plotkin, C. Stein, and É. Tardos. Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts. *SIAM J. Comput.*, 23(3):466–487, 1994.

[20] R. Krauthgamer and Y. Rabani. Improved lower bounds for embeddings into l1. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1010–1017, New York, NY, USA, 2006. ACM Press.

[21] K. Lang and S. Rao. Finding near-optimal cuts: an empirical evaluation. In *SODA '93: Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, pages 212–221, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics.

[22] F. T. Leighton, F. Makedon, S. A. Plotkin, C. Stein, É. Stein, and S. Tragoudas. Fast approximation algorithms for multicommodity flow problems. *J. Comput. Syst. Sci.*, 50(2):228–243, 1995.

[23] T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 422–431, 1988.

[24] T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, 1999.

[25] T. Leong, P. Shor, and C. Stein. Implementation of a combinatorial multicommodity flow algorithm. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science Volume 12: Network flows and matching*, pages 387–406. DIMACS, 1991.

[26] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.

[27] F. Shahrokhi and D. W. Matula. The maximum concurrent flow problem. *J. ACM*, 37(2):318–334, 1990.

[28] D. B. Shmoys. Cut problems and their applications to divide-and-conquer. In D. S. Hochbaum, editor, *Approximation algorithms for NP-hard problems*, pages 192–235. PWS Publishing Company, 1997.