Research Statement

Chao Xu

My main research interests are in the area of combinatorial optimization. In the past, I have worked on cuts and connectivity problems in hypergraphs, and global and fixed-terminal cut problems in graphs. Recently, I have been trying to explore the implication of my work to CSP theory, submodular function optimization and matroid rank reduction. I would like to branch out into understanding the tractability issues in those areas. I also have interests in classic algorithmic problems, and would like to reinvest in the area of computational geometry.

Past works

My earlier work is in the field of computational geometry. In collaboration with Chang and Erickson, I showed a fast algorithm for recognizing a generalization of simple polygons [7]. I have taken up more interest in combinatorial structures after I made progress in element connectivity algorithms with Chekuri and Rukkanchanunt [8]. Here I present some of my more recent works.

Fast algorithms for hypergraph cuts

A hypergraph generalizes a graph by allowing each edge to contain more than 2 vertices. A cut is a set of edges, such that its removal disconnects the hypergraph. A min-cut is a cut with the smallest number of edges. Richer models present challenges in designing fast algorithms due to their additional complexity. For example, the *size* of a hypergraph is the sum of the edge sizes, which can be much larger than the number of edges. I worked on finding fast algorithms in hypergraphs pertaining to cuts and connectivity. My hypergraph algorithms match the state of the art graph algorithms, and sometimes they are conceptually simpler. In the joint work with Chekuri, I showed the following results [9, 10].

Finding all min-cuts. The cactus representation captures all min-cuts of a graph [17]. Finding all min-cuts is equivalent to computing the cactus representation. It took many years of continuous progress to reach the fastest time and smallest space algorithm [20, 24, 43, 44, 45]. Finding a cactus representation takes the same amount of time as finding a *single min-cut*, and the space complexity is *linear*. A hypercactus representation captures all min-cut information of a hypergraph [12, 19]. Finding the hypercactus representation takes polynomial time, but it is much slower than finding a min-cut.

We show that finding all min-cuts of a hypergraph is no harder than finding a single min-cut. The algorithm is much simpler than the graph counterpart by applying the conceptually clean Cunningham's decomposition framework [14]. Finally, the algorithm takes optimal *linear* space. The framework depends on finding *splits*, min-cuts that separate at least 2 vertices on each side. The approach alone is already prohibitive: finding a split is no easier than finding a min-cut. Our main algorithmic insight is a near-linear time split oracle. The oracle either finds a split or gives us two vertices whose contraction does not destroy any min-cut.

Cut sparsifiers. Storing a dense graph in memory is expensive. It is unavoidable if we want to access all cut values exactly. If we want an approximate value, then we can greatly decrease the number of edges in the storage. A sparse subgraph that preserves all cuts to within a $(1 \pm \epsilon)$ factor is a cut sparsifier. A near-linear time algorithm can find an $O(n \log n)$ edge cut sparsifier in an n-vertex graph. The algorithm samples edges by an appropriate distribution [3]. A spectral sparsifier is a generalization of a cut sparsifier, and there exists one with $O(n)$ edges [40].

A cut sparsifier for a hypergraph also exists [26, 37]. The same sampling algorithm works. However, finding the probability distribution is the bottleneck. The sampling probability is inversely proportional to the strength of an edge. The strength measures the importance of an edge to the cuts that it crosses. We provided a near-linear time algorithm to approximate the strength. As a consequence, we get a near-linear time algorithm for a hypergraph cut sparsifier. The main tool is a fast algorithm for k-certificates, as we will see next.

k-certificates. A graph is k-edge-connected if removing any $k-1$ edges does not disconnect the graph. 1-edge-connectivity is the standard connectivity. Every connected graph contains a spanning tree that *certifies* the connectivity of a graph. A k-edge-connected graph has a similar $O(kn)$ edge subgraph, a k-certificate, that certifies that the graph is k-edge-connected. k-certificates are powerful in algorithm design because it is a sparse graph that demonstrates a lower bound on the connectivity. Most importantly, finding a k-certificate takes *linear time* [42]. Therefore, finding a k-certificate is a preprocessing step in various graph algorithms [23]. A k-certificate also exists for a k-edge-connected hypergraph [26]. To find a k-certificate of a hypergraph, one repeatedly strips off 1-certificates. Unfortunately, it is too slow for large k, and it does not work for weighted graphs.

We show that a k-certificate for a hypergraph can also be found in *linear time*. Moreover, the total *size* of the k-certificate we find is $O(kn)$, not just the number of edges. In unweighted settings, the result gives us faster algorithms for finding min-cuts, approximate min-cuts and max flows. In weighted settings, k-certificates are essential in finding cut sparsifiers.

Minimum k-cut in hypergraphs

A set of edges is a k-cut if there are at least k components after removing them from the graph. A 2-cut is a standard cut. Goldschmidt and Hochbaum made a surprising discovery that a min k-cut in graphs can be found in polynomial time [25]. Subsequent works improved the running time using techniques including divide and conquer, tree packing, and randomized contractions [30, 31, 33, 48]. Finding a min k-cut of a hypergraph has applications in network reliability and clustering in VLSI design [1, 51]. It is much harder to find a min k-cut in a hypergraph than in a graph. First, the uncrossing observation of graph k-cuts in Goldschmidt and Hochbaum does not apply to hypergraphs. Second, the tree packing technique fails in the presence of hyperedges that contain a large number of vertices. The known algorithms tackle only severely restricted special cases. Xiao gave an algorithm for finding a min 3-cut and pointed out some fundamental difficulties in extending it to 4-cuts [50]. Fukunaga gave a polynomial time algorithm when the edges have constant size [21]. Solving the hypergraph k-cut problem remained an open problem since the works of Goldschmidt and Hochbaum (1994).

Collaborating with Chandrasekaran and Yu, we resolve this problem by a randomized polynomial time algorithm for min k-cut in hypergraphs [6]. The algorithm is based on Karger's randomized contraction algorithm for graph cuts [32]. The algorithm repeatedly samples an edge with appropriate probability and contracts. When few vertices remain, it outputs a random k-cut. The difficulty lies in finding a non-trivial probability distribution where the algorithm works. Our algorithm actually works in a more general setting. It can find the minimum k-cut for hedgegraphs with constant span, which is a generalization of hypergraphs.

Global and fixed-terminal cut problems

The k-way cut problem, the *fixed-terminal* version of k-cut, asks one to find a k-cut that separates a given set of k terminals. The 2-way cut problem is the *st*-min-cut problem, which is solvable by a maximum flow computation. Unfortunately, 3-way cut is already NP-hard [15]. In contrast, finding a min k-cut that separates *some* k terminals is solvable in polynomial time. The cut problems without fixed terminals are global cut problems.

The curious complexity gap between fixed-terminal cut problems and global cut problems is a well-known phenomenon in graphs. Collaborating with Bérczi, Chandrasekaran, Király and Lee, I showed some initial results in this space for directed graphs [4]. Specifically, the global problem is *strictly easier* than the fixed-terminal problem in both node and edge based variants. There is also the semi-global cut problem, where one can fix fewer terminals than the number of terminals to be separated. One particular question is finding a k-cut that separate terminals in T. It is hard when $|T| \geq 3$, polynomial time solvable for $|T| \leq 1$ and nothing was known for $|T| = 2$ [27]. In the same paper, I resolve the final case by giving a polynomial time algorithm. This suggests the problems in this space lie on the very thin line between intractability and tractability.

The various global and fixed-terminal cut problems have a common generalization, the minimum violation problem. Let the violation of a vertex map between two graphs be the number of edges not mapped to an edge. The k-cut problems can be seen as finding a surjective vertex map from G to H with minimum violation, where H consists of k isolated self-loops. A k-way cut is equivalent to a vertex map where the terminal vertices are fixed in the mapping. A graph H is r-tractable or s-tractable if finding the minimum violation fixed-terminal vertex map or minimum violation surjective vertex map is tractable, respectively. Deciding if a graph is r-tractable is a special case of valued constraint satisfaction problem (VCSP). The VCSP dichotomy theorem implies an *algebraic* dichotomy theorem of r-tractable graphs [39]. However, the algebraic criterion is not efficiently testable. In collaboration with Kawarabayashi, I found a *combinatorial* dichotomy theorem and a polynomial time recognition algorithm for r-tractable graphs [35]. There is some initial progress on s-tractable graphs, including the case when the graph is a disjoint union of s-tractable graphs. As a consequence, we found the first deterministic algorithm for size-constrained k-cut problem [35].

The subset sum problem

The subset sum problem takes a set of n natural numbers and a target number t. It outputs if there exists a subset of elements that sum to t. The subset sum problem is one of Karp's 21 NP-complete problems [34]. A *faster pseudopolynomial time* algorithm for the subset sum problem leads to *faster polynomial time* algorithms for many combinatorial problems [18, 28]. The dynamic programming $O(nt)$ time algorithm by Bellman stood unchallenged for 60 years [2], except for a log factor improvement by packing the dynamic programming table into words [47].

Collaborating with fellow PhD student Koiliaris, I gave an $\tilde{O}(\sqrt{n}t)$ time algorithm for the subset sum problem [38]. It is the first polynomial factor improvement in 60 years. It rapidly became my most cited paper. The algorithm is a fast divide and conquer algorithm that cleverly discards unnecessary information. Moreover, I considered the subset sum problem where the numbers are integers modulo m. Surprisingly, in this case, our algorithm leads to an improvement in codes correcting limited magnitude errors, a problem in error correction codes [11].

Future research

My specialty has lead me to an advantageous position in the intersection of multiple areas, including CSP theory, matroid theory and submodular functions. A starting point toward a unified understanding of tractability in these domains is to answer why min k-cut is tractable. I demonstrate a few initial directions in the near term. Certainly, more will be uncovered during the investigation.

Theory of surjective valued CSPs. VCSP is a powerful modeling tool that can model all fixed-terminal graph cuts problems. The recent resolution of the CSP dichotomy conjecture [5, 52] brings a conclusion to a long search, and it gives an *algebraic criterion* to the tractability of VCSP [39]. However, there are more challenges. First, the dichotomy result for VCSP only works for constraints with *constant* size. In particular, VCSP does not offer any insights towards hypergraph cut problems. Second, VCSP is fundamentally incapable of modeling global cut problems. To handle global cut problems, we need to study surjective VCSP (SVCSP) with arbitrarily large constraints. SVCSP is becoming the next central question for the CSP community. The boolean case with constant size constraints was solved only this year [22]. My results on global cuts and hypergraph cuts is a beginning of exchanges with the CSP community, and I want to contribute to the development of the SVCSP theory.

Submodular k-partition. A submodular function is the set function analog of a convex function. It captures the notion of diminishing returns, and presents itself in various economics, machine learning, and optimization topics. In the submodular k-partition problem, we want to partition the groundset into k parts to minimize the sum of their value under the submodular function. In the graph case, the submodular function is the cut function. There is an intricate connection between submodularity and VCSP. For example, in the special case of MAXCSP, tractability is determined completely by submodularity [16]. A minimum submodular 3-partition can be found in polynomial time [41]. It is unclear if submodular k-partition can be solved for $k \geq 4$, but my hypergraph k-cut result suggests a positive result is likely. If finding a minimum k-partition takes polynomial time, then it explains why hypergraph k-cut is tractable, and potentially contributes to SVCSP theory.

Matroid rank k-reduction. A matroid is a combinatorial structure that captures the idea of linear independence. Given a matroid, we remove the smallest set of elements to decrease the rank of the matroid by at least k. Finding such a set is the k-reduction problem. This problem is closely related to matroid interdiction [13]. The graph k-cut problem is precisely the $(k-1)$-reduction problem on graphic matroids. The

1-reduction problem is equivalent to the cogirth problem, which is already NP-hard on binary matroids [49]. However, cogirth is easy to compute for many matroids, including hypergraphic, regular and transversal matroids [36,46]. As one can see from the graph example, k-reduction is highly non-trivial and naive attempts like applying 1-reduction k times do not work. Despite the fact that the k-reduction problem was mentioned as an open problem by Goldschmidt and Hochbaum in 1994, there are very few results. Only recently, the k-reduction problem was shown to be tractable for partition matroids [29]. The conjecture is that a minor-closed class of matroids where 1-reduction is tractable, k-reduction is also tractable. The goal is to verify if they are indeed equivalent under polynomial time reduction.

In the long term, I am open to exploring other areas in combinatorial optimization and its interaction with other fields. Specifically, I would like to look into its connection to problems in computational geometry.

References

[1] C. J. Alpert and A. B. Kahng. Recent directions in netlist partitioning: A survey. *Integr. VLSI J.*, 19(1-2):1–81, Aug 1995.

[2] R. Bellman. Notes on the theory of dynamic programming IV - maximization over discrete sets. *Naval Research Logistics Quarterly*, 3(1-2):67–70, 1956.

[3] A. A. Benczúr and D. R. Karger. Randomized approximation schemes for cuts and flows in capacitated graphs. *SIAM J. Comput.*, 44(2):290–319, 2015. Preliminary versions appeared in STOC '96 and SODA '98.

[4] K. Bérczi, K. Chandrasekaran, T. Király, E. Lee, and C. Xu. Global and Fixed-Terminal Cuts in Digraphs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2017)*, volume 81 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:20, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[5] A. A. Bulatov. A dichotomy theorem for nonuniform CSPs. In *Proc. FOCS 2017*, 2017.

[6] K. Chandrasekaran, C. Xu, and X. Yu. Hypergraph k-Cut in Randomized Polynomial Time. *To appear in SODA '18*, 2018.

[7] H.-C. Chang, J. Erickson, and C. Xu. Detecting weakly simple polygons. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1655–1670. SIAM, 2015.

[8] C. Chekuri, T. Rukkanchanunt, and C. Xu. On element-connectivity preserving graph simplification. In *Algorithms – ESA 2015*, volume 9294 of *Lecture Notes in Computer Science*, pages 313–324. Springer Berlin Heidelberg, 2015.

[9] C. Chekuri and C. Xu. Computing minimum cuts in hypergraphs. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1085–1100.

[10] C. Chekuri and C. Xu. A note on approximate strengths of edges in a hypergraph. *CoRR*, abs/1703.03849, 2017.

[11] Z. Chen, I. E. Shparlinski, and A. Winterhof. Covering sets for limited-magnitude errors. *IEEE Transactions on Information Theory*, 60(9):5315–5321, Sep 2014.

[12] E. Cheng. Edge-augmentation of hypergraphs. *Mathematical Programming*, 84(3):443–465, Apr 1999.

[13] S. R. Chestnut and R. Zenklusen. Interdicting structured combinatorial optimization problems with 0, 1-objectives. *Mathematics of Operations Research*, 42(1):144–166, 2017.

[14] W. H. Cunningham. Decomposition of submodular functions. *Combinatorica*, 3(1):53–68, 1983.

[15] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The Complexity of Multiterminal Cuts. *SIAM Journal on Computing*, 23(4):864–894, Aug. 1994.

[16] V. Deineko, P. Jonsson, M. Klasson, and A. Krokhin. The approximability of MAX CSP with fixed-value constraints. *J. ACM*, 55(4):16:1–16:37, Sep 2008.

[17] M. V. L. Efim A. Dinits, Alexander V. Karzanov. On the structure of a family of minimal weighted cuts in graphs. In *Studies in Discrete Mathematics*, pages 290–306. Nauka (Moskva), 1976.

[18] D. Eppstein. Minimum range balanced cuts via dynamic subset sums. *Journal of Algorithms*, 23(2):375 – 385, 1997.

[19] T. Fleiner and T. Jordán. Coverings and structure of crossing families. *Mathematical Programming*, 84(3):505–518, Apr 1999.

[20] L. Fleischer. Building chain and cactus representations of all minimum cuts from Hao–Orlin in the same asymptotic run time. *Journal of Algorithms*, 33(1):51 – 72, 1999.

[21] T. Fukunaga. Computing minimum multiway cuts in hypergraphs. *Discrete Optimization*, 10(4):371 – 382, 2013.

[22] P. Fulla, H. Uppman, and S. Živný. The complexity of boolean surjective general-valued CSPs. *CoRR*, abs/1702.04679, 2017.

[23] H. N. Gabow. A Matroid Approach to Finding Edge Connectivity and Packing Arborescences. *Journal of Computer and System Sciences*, 50(2):259–273, Apr 1995.

[24] H. N. Gabow. The minset-poset approach to representations of graph connectivity. *ACM Trans. Algorithms*, 12(2):24:1–24:73, Feb 2016.

[25] O. Goldschmidt and D. S. Hochbaum. A Polynomial Algorithm for the k-cut Problem for Fixed k. *Mathematics of Operations Research*, 19(1):24–37, 1994.

[26] S. Guha, A. McGregor, and D. Tench. Vertex and hyperedge connectivity in dynamic graph streams. In *Proceedings of the 34th ACM Symposium on Principles of Database Systems*, PODS '15, pages 241–247, New York, NY, USA, 2015. ACM.

[27] F. Guiñez and M. Queyranne. The size-constrained submodular k-partition problem. Manuscript, https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxmbGF2aW9ndWluZXpob21lcGFnZXxneDo0NDVlMThkMDg4ZWRlOGI1, 2012.

[28] D. S. Hochbaum and A. Pathria. The bottleneck graph partition problem. *Networks*, 28(4):221–225, 1996.

[29] G. Joret and A. Vetta. Reducing the rank of a matroid. *Discrete Mathematics & Theoretical Computer Science*, Vol. 17 no.2, Sept. 2015.

[30] Y. Kamidoi, S. Wakabayashi, and N. Yoshida. A Divide-and-Conquer Approach to the Minimum k-Way Cut Problem. *Algorithmica*, 32(2):262–276, Feb. 2002.

[31] Y. Kamidoi, N. Yoshida, and H. Nagamochi. A deterministic algorithm for finding all minimum k-way cuts. *SIAM J. Comput.*, 36(5):1329–1341, 2006.

[32] D. R. Karger. *Random Sampling in Graph Optimization Problems*. PhD thesis, Stanford University, Feb 1995.

[33] D. R. Karger and C. Stein. A new approach to the minimum cut problem. *Journal of the ACM (JACM)*, 43(4):601–640, 1996.

[34] R. M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972.

[35] K. Kawarabayashi and C. Xu. Minimum violation vertex maps and their application to cut problems. Unpublished Manuscript., 2017.

[36] T. Király. Computing the minimum cut in hypergraphic matroids. Technical Report QP-2009-05, Egerváry Research Group, Budapest, 2009. www.cs.elte.hu/egres.

[37] D. Kogan and R. Krauthgamer. Sketching cuts in graphs and hypergraphs. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, ITCS '15, pages 367–376, New York, NY, USA, 2015. ACM.

[38] K. Koiliaris and C. Xu. A faster pseudopolynomial time algorithm for subset sum. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, pages 1062–1072. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2017.

[39] V. Kolmogorov, A. Krokhin, and M. Rolinek. The complexity of general-valued CSPs. In *Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, FOCS '15, pages 1246–1258, Washington, DC, USA, 2015. IEEE Computer Society.

[40] Y. T. Lee and H. Sun. Constructing linear-sized spectral sparsification in almost-linear time. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 250–269. IEEE, 2015.

[41] H. Nagamochi. Approximating Minimum k-partitions in Submodular Systems. In *Second International Conference on Informatics Research for Development of Knowledge Society Infrastructure (ICKS'07)*, pages 119–128. IEEE, Jan. 2007.

[42] H. Nagamochi and T. Ibaraki. A linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph. *Algorithmica*, 7(1-6):583–596, 1992.

[43] H. Nagamochi and T. Kameda. Constructing cactus representation for all minimum cuts in an undirected network. *Journal of the Operations Research Society of Japan*, 39(2):135–158, 1996.

[44] H. Nagamochi, S. Nakamura, and T. Ishii. Constructing a cactus for minimum cuts of a graph in $O(mn+n^2 \log n)$ time and $O(m)$ space. *IEICE Transactions on Information and Systems*, E86-D(2):179–185, 2003.

[45] H. Nagamochi, Y. Nakao, and T. Ibaraki. A fast algorithm for cactus representations of minimum cuts. *Japan Journal of Industrial and Applied Mathematics*, 17(2):245, Jun 2000.

[46] F. Panolan, M. S. Ramanujan, and S. Saurabh. *On the Parameterized Complexity of Girth and Connectivity Problems on Linear Matroids*, pages 566–577. Springer International Publishing, 2015.

[47] D. Pisinger. Dynamic programming on the word RAM. *Algorithmica*, 35(2):128–145, 2003.

[48] M. Thorup. Minimum k-way cuts via deterministic greedy tree packing. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 159–166. ACM, 2008.

[49] A. Vardy. The intractability of computing the minimum distance of a code. *IEEE Trans. Inf. Theor.*, 43(6):1757–1766, Nov 1997.

[50] M. Xiao. Finding minimum 3-way cuts in hypergraphs. *Information Processing Letters*, 110(14-15):554–558, 2010. Preliminary version in TAMC 2008.

[51] L. Zhao. *Approximation algorithms for partition and design problems in networks.* PhD thesis, Graduate School of Informatics, Kyoto University, Japan, 2002.

[52] D. Zhuk. A proof of CSP dichotomy conjecture. In *Proc. FOCS 2017*, 2017.