

Randomized Algorithms for Graph Problems

Warren Schudy

October 26, 2007

Randomization is a powerful tool in the design of algorithms. In this proposal we use randomness to make progress on several old and fundamental problems. We *close the approximability* of feedback arc set on tournament graphs, first studied around 1961 [16], by designing a PTAS. We give a practical algorithm with *improved performance* for finding strongly connected components in parallel (first studied around 1988 [6]). We also find and analyze *more elegant algorithms* for max cut of dense graphs (1995). Additional details on these results are available in our papers (see <http://www.cs.brown.edu/~ws/>).

1 Strongly Connected Components in Parallel [15]

Breadth-first and depth-first search have many applications in the analysis of directed graphs. Breadth-first search can be used to compute the vertices that are reachable from a given vertex and directed spanning trees. Depth-first search can additionally determine if a graph is acyclic, topologically sort an acyclic graph and compute strongly connected components (SCCs). Huge graphs such as the web graph can only be processed effectively in parallel, but efforts to parallelize these algorithms have met with mixed success.

In this project we focus on strongly connected components and topological sort. Both these problems can be solved in polylog time with $n^{2.38}$ processors using matrix-exponentiation techniques to compute the transitive closure of the graph [6]. Spencer [17] gives algorithms that trade-off

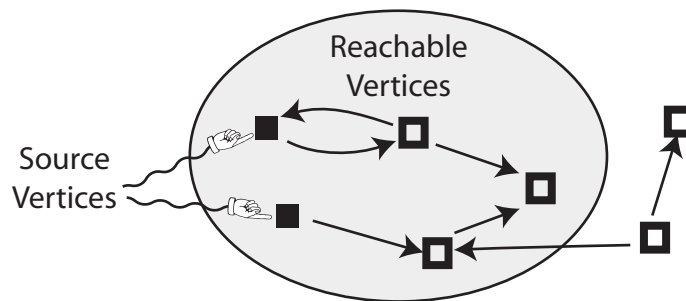


Figure 1: Illustration of a reachability query.

time and work, with runtime $\tilde{O}(n/p^{1/3})$ on p processors. Coppersmith, Fleischer, Hendrickson and Pinar [5] give a practical parallel divide and conquer algorithm for computing SCCs using reachability queries (see Figure 1). They prove $O(m \log n)$ serial runtime, but do not prove that extra processors reduce the runtime. Their algorithm has been implemented in the context of scientific computing [14].

Building on ideas from [5], we give a simple parallel randomized algorithm for the topological sort and strongly connected components problems with runtime $O(\tau \log^2 n)$, where τ is the runtime for a reachability query. On sparse graphs, which are the most important ones in practice, this improves the best-known number of processors needed from the cube of the speedup [17] to the square.

2 Feedback Arc Set and Rank Aggregation [12]

For general directed graphs, the feedback arc set (FAS) problem consists of ordering the vertices so as to reverse the fewest number of edges (see Figure 2). Unfortunately the problem is NP-hard to approximate better than 1.36.

A *tournament* is the special case of directed graphs where every pair of vertices is connected by exactly one of the two possible directed edges. The FAS problem for tournaments has a long history, starting in the early 1960s in combinatorics and statistics [16]. Unfortunately, even on tournaments, the FAS problem is still NP-hard [1, 2]. The best previously known approximation algorithms achieved constant factor approximations [1]. Our main result is a randomized *approximation scheme*; that is, for any $\epsilon > 0$, an algorithm that achieves at most $1 + \epsilon$ times the optimal number of reversed arcs. The runtime is polynomial in the number of vertices, i.e. it is a polynomial time approximation scheme (PTAS). (The runtime is doubly exponential in $1/\epsilon$, so our algorithm is unfortunately not practical without heuristic modifications.)

An important application is the NP-hard *Kemeny Young rank aggregation* [11] problem, illustrated by example in Figure 2. This form of rank aggregation has desirable properties for application

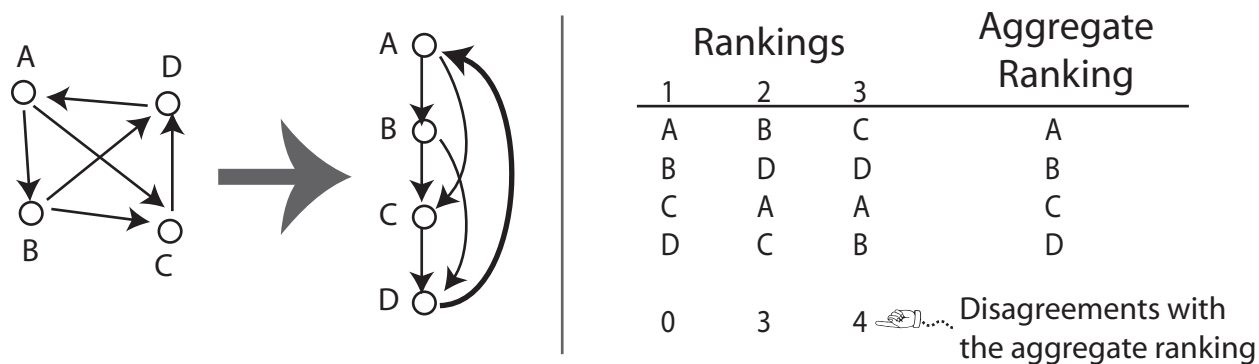


Figure 2: Feedback arc set tournament (left) and Kemeny-Young rank aggregation (right) problems.

in search engine aggregation [8]. The previously best known approximation factor is $4/3$ [1], but we give a polynomial-time approximation scheme.

These PTASs close the approximability of these problems: an approximation factor of 1 is not possible by NP-hardness, and we give approximation algorithms for every factor greater than 1. Both problems date from the days of quicksort, long before NP-hardness was discovered!

Allowing the input rankings to contain ties yields a generalization called *partial rank aggregation*. Previous results for Kemeny-Young rank aggregation have generalized to partial rank aggregation, so we are investigating how to generalize our result as well.

3 Maximum Cut [13]

The NP-hard MaxCut problem strives to partition the vertices of an input graph into two parts to maximize the number of edges across the cut. There has been much previous work on designing approximation schemes for dense graph instances of MaxCut, and some generalizations, see for example [3, 10, 4, 7, 9]. There are many algorithms, each with its own unique strong points, based on a combinatorial approach, or on spectral techniques, or on smoothed linear programming relaxations. This suggests that the problem is “easy” in some sense, and thus simple algorithms have a chance to be successful as well.

The greedy algorithm for MaxCut, depicted in Figure 3, considers vertices one by one in arbitrary order and places each of them on the left or right side of the cut, depending on the number of neighbors that are already placed on each side. It is well known that the greedy algorithm is a 2-approximation. We take advantage of the power of randomness by considering vertices *in random order*. See Algorithm 1.



Figure 3: Illustration of the greedy algorithm for max cut

Algorithm 1 Randomized greedy maxcut algorithm

Repeat a number of times that depends only on ϵ :

 For each vertex v of V in random order,

 Greedly place v on the side of the cut that maximizes the number of cut edges

Output the best cut found

Our main result is a martingale-heavy proof that this very simple algorithm is an approximation scheme on dense graphs. See [13] for extensions to constraint satisfaction problems and sample complexity.

References

- [1] N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: ranking and clustering. In *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 684–693, New York, NY, USA, 2005. ACM Press.
- [2] N. Alon. Ranking tournaments. *SIAM J. Discret. Math.*, 20(1):137–142, 2006.
- [3] N. Alon, W. F. de la Vega, R. Kannan, and M. Karpinski. Random sampling and approximation of max-csps. *J. Comput. Syst. Sci.*, 67(2):212–243, 2003.
- [4] S. Arora, D. Karger, and M. Karpinski. Polynomial time approximation schemes for dense instances of NP-hard problems. In *STOC '95: Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 284–293, 1995.
- [5] D. Coppersmith, L. Fleischer, B. Hendrickson, and A. Pinar. A divide-and-conquer algorithm for identifying strongly connected components. Technical Report RC23744, IBM Research, 2005.
- [6] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. In *STOC '87: Proceedings of the nineteenth annual ACM Symposium on Theory of Computing*, pages 1–6, New York, NY, USA, 1987. ACM Press.
- [7] W. F. de la Vega. Max-cut has a randomized approximation scheme in dense graphs. *Random Struct. Algorithms*, 8(3):187–198, 1996.
- [8] C. Dwork, S. R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *World Wide Web*, pages 613–622, 2001.
- [9] A. M. Frieze and R. Kannan. Quick approximation to matrices and applications. *Combinatorica*, 19(2):175–220, 1999.
- [10] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998.
- [11] J. Kemeny and J. Snell. *Mathematical Models in the Social Sciences*. Blaisdell, New York, 1962. Reprinted by MIT press, Cambridge, 1972.
- [12] C. Kenyon-Mathieu and W. Schudy. How to rank with few errors. In *STOC '07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 95–103, New York, NY, USA, 2007. ACM Press.

- [13] C. Mathieu and W. Schudy. Yet another algorithm for dense max cut: Go greedy. In *SODA '08: Proceedings of the nineteenth annual ACM-SIAM Symposium on Discrete Algorithms*, 2008.
- [14] W. McLendon, III, B. Hendrickson, S. J. Plimpton, and L. Rauchwerger. Finding strongly connected components in distributed graphs. *J. Parallel Distrib. Comput.*, 65(8):901–910, 2005.
- [15] W. Schudy. Strongly connected components in $O(\log^2 n)$ reachability queries. Manuscript in preparation for STOC 2008, available from <http://www.cs.brown.edu/~ws/scc.pdf>.
- [16] P. Slater. Inconsistencies in a schedule of paired comparisons. *Biometrika*, 48:303–312, 1961.
- [17] T. H. Spencer. Time-work tradeoffs for parallel algorithms. *J. ACM*, 44(5):742–778, 1997.