Randomized Parallel Approximations to Max Flow, (1991; Serna, Spirakis)

Maria Serna, Technical University of Catalonia, www.cs.upc.edu/~mjserna

INDEX TERMS: Maximum flow, Maximum Weight Matching, Randomized algorithm, Approximation, RNC class, FNCAS, PRAM algorithm.
SYNONIMS: Approximate Maximum Flow construction.

1 PROBLEM DEFINITION

The work of Serna and Spirakis provides a parallel approximation schema for the Maximum flow problem. An approximate algorithm provides a solution whose cost is within a factor of the optimal solution. The notation and definitions are the standard ones for networks and flows (see for example [7], [2]).

A *network* $N = (G, s, t, c)$ is a structure consisting of a directed graph $G = (V, E)$, two distinguished vertices, $s, t \in V$ (called the *source* and the *sink*), and $c : E \to \mathbb{Z}^+$, an assignment of an integer capacity to each edge in E. A *flow function* f is an assignment of a nonnegative number to each edge of G (called the flow into the edge) such that first at no edge does the flow exceed the capacity, and second for every vertex except s and t, the sum of the flows on its incoming edges equals the sum of the flows on its outgoing edges. The *total flow* of a given flow function f is defined as the net sum of flow into the sink t. The Maximum flow problem can be stated as

Name Maximum flow
Input A network $N = (G, s, t, c)$
Output Find a flow f for N for which the total flow is maximum.

Maximum Flows and Matchings The Maximum Flow Problem is closely related to the Maximum Matching problem on bipartite graphs.

Given a graph $G = (V, E)$ and a set of edges $M \subseteq E$ is a *matching* if in the subgraph (V, M) all vertices have degree at most one. A *maximum matching* for G is a matching with maximum number of edges. For a graph $G = (V, E)$ with weight $w(e)$, the *weight* of a matching M is the sum of the weights of the edges in M. The problem can be stated as follows:

Name Maximum Weight Matching
Input A graph $G = (V, E)$ and a weight $w(e)$ for each edge $e \in E$
Output Find a matching of G with the maximum possible weight.

There is a standard reduction from the Maximum Matching problem for bipartite graphs to the Maximum Flow problem ([7], [8]). In the general weighted case one have just to look at each edge with capacity $c > 1$ as c edges joining the same points each with capacity one, and transform the multigraph obtained as shown before. Notice that to perform this transformation it is required a c value which is polynomially bounded. The whole procedure was introduced by Karp, Upfal and Wigderson [5] providing the following results

Theorem 1. *The* Maximum Matching *problem for bipartite graphs is* NC *equivalent to the* Maximum Flow *problem on networks with polynomial capacities. Therefore, Maximum Flow with polynomial capacities problem belongs to the class RNC.*

2 KEY RESULTS

The first contribution is an extension of Theorem 1 to a generalization of the problem, namely the Maximum Flow on networks with polynomially bounded maximum flow. The proof is based on the construction (in NC) of a second network which has the same maximum flow but for which the maximum flow and the maximum capacity in the network are polynomially related.

Lemma 2. *Let $N = (G, s, t, c)$. Given any integer k, there is an* NC *algorithm that decides whether $f(N) \geq k$ or $f(N) < k\,m$.*

Since Lemma 2 applies even to numbers that are exponential in size, they get

Lemma 3. *Let $N = (G, s, t, c)$ be a network. there is an* NC *algorithm that computes an integer value k such that $2^k \leq f(N) < m\ 2^{k+1}$.*

The following lemma establishes the NC-reduction from the Maximum Flow problem with polynomial maximum flow to the Maximum Flow problem with polynomial capacities.

Lemma 4. *Let $N = (G, s, t, c)$ be a network, there is an* NC *algorithm that constructs a second network $N_1 = (G, s, t, c_1)$ such that*

$$\log(Max(N_1)) \leq \log(f(N_1)) + O(\log n)$$

and $f(N) = f(N_1)$.

Lemma 4 shows that the Maximum Flow problem restricted to networks with polynomially bounded maximum flow is NC-reducible to the Maximum Flow problem restricted to polynomially bounded capacities, the latter problem is a simplification of the former one, so the following results follows.

Theorem 5. *For each polynomial p, the problem of constructing a maximum flow in a network N such that $f(N) \leq p(n)$ is* NC*-equivalent to the problem of constructing a maximum matching in a bipartite graph, and thus it is in* RNC*.*

Recall that [5] gave us an $O(\log^2 n)$ randomized parallel time algorithm to compute a maximum matching. The combination of this with the reduction from the Maximum Flow problem to the Maximum Matching leads to the following result.

Theorem 6. *There is a randomized parallel algorithm to construct a maximum flow in a directed network, such that the number of processors is bounded by a polynomial in the number of vertices and the time used is $O((\log n)^\alpha \log f(N))$ for some constant $\alpha > 0$.*

The previous theorem is the first step towards finding a approximate maximum flow in a network N by an RNC algorithm. The algorithm, given N and an $\epsilon > 0$, outputs a solution f' such that $f(N)/f' \leq 1 + 1/\epsilon$. The algorithm uses a polynomial number of processors (independent of ϵ) and parallel time $O(\log^\alpha n(\log n + \log \epsilon))$, where α is independent of ϵ. Thus the algorithm is an RNC one as long as ϵ is at most polynomial in n. (Actually ϵ can be $O(n^{\log^\beta n})$ for some β.) Thus being a Fully RNC approximation scheme (FRNCAS).

The second ingredient is a rough NC approximation to the Maximum Flow problem.

Lemma 7. *Let $N = (G, s, t, c)$ be a network. Let $k \geq 1$ be an integer, then there is an* NC *algorithm to construct a network $M = (G, s, t, c_1)$ such that $k\, f(M) \leq f(N) \leq k\, f(M) + km$.*

Putting all together and allowing randomization the algorithm can be sketched as follows:

FAST-FLOW($N = (G, s, t, c), \epsilon$)

1. Compute k such that $2^k \leq F(N) \leq 2^{k+1}m$.
2. Construct a network N_1 such that
$$\log(Max(N_1)) \leq \log(F(N_1)) + O(\log n).$$
3. If $2^k \leq (1+\epsilon)m$ then $F(N) \leq (1+\epsilon)m^2$ so use Algorithm **??** to solve the Maximum Flow problem in N as a Maximum Matching and **return**
4. Let $\beta = \left\lfloor \dfrac{2^k}{(1+\epsilon)m} \right\rfloor$. Construct N_2 from N_1 and β using the construction in Lemma 7.
5. Solve the Maximum Flow problem in N_2 as a Maximum Matching.
6. Output $F' = \beta F(M_2)$ and for all $e \in E$, $f'(e) = \beta f(e)$.

Theorem 8. *Let $N = (G, s, t, c)$ be a network. Then, algorithm FAST-FLOW is an* RNC *algorithm such that for all $\epsilon > 0$ at most polynomial in the number of network vertices, the algorithm computes a legal flow of value f' such that*

$$\frac{f(N)}{f'} \leq 1 + \frac{1}{\epsilon}.$$

Furthermore, the algorithm uses a polynomial number of processors and runs in expected parallel time $O(\log^\alpha n(\log n + \log \epsilon))$, for some constant α, independent of ϵ.

3 APPLICATIONS

The *rounding/scaling* technique is used in general to deal with problems that are hard due to the presence of large weights in the problem instance. The technique modifies the problem instance in order to produce a second instance that has no large weights, and thus can be solved efficiently. The way in which a new instance is obtained consists in computing first an estimate of the optimal value (when needed) in order to discard unnecessary high weights. Then the weights are modified, scaling them down by an appropriate factor that depends on the estimation and the allowed error. The rounding factor is determined in such a way that the so obtained instance can be solved efficiently. Finally a last step consisting in scaling up the value of the "easy" instance solution is performed in order to meet the corresponding accuracy requirements.

It is known that in the sequential case, the only way to construct FPTAS uses rounding/scaling and interval partition [6]. In general, both techniques can be parallelized, although sometimes the details of the parallelization are non-trivial [1].

The Maximum Flow problem has a long history in computer Science. Here are recorded some results about its parallel complexity. Goldschlager, Shaw and Staples showed that the Maximum Flow problem is P-complete [3]. The P-completeness proof for Maximum Flow uses large capacities on the edges; in fact the values of some capacities are exponential in the number of network vertices. If the capacities are constrained to be no greater than some polynomial in the number of network vertices the problem is in ZNC. In the case of planar networks it is known that the Maximum Flow problem is in NC, even if arbitrary capacities are allowed [4].

4 OPEN PROBLEMS

The parallel complexity of the Maximum Weight Matching problem when the weight of the edges are given in binary is still an open problem. However, as mentioned earlier, there is a randomized NC algorithm to solve the problem in $O(\log^2 n)$ parallel steps, when the weights of the edges are given in unary. The scaling technique has been used to obtain fully randomized NC approximation schemes, for the Maximum Flow and Maximum Weight Matching problems (see [10]). The result appears to be the best possible in regard of full approximation, in the sense that the existence of an FNCAS for any of the problems considered is equivalent to the existence of an NC algorithm for perfect matching which is also still an open problem.

5 EXPERIMENTAL RESULTS

None is reported.

6 DATA SETS

None is reported.

7 URL to CODE

None is reported.

8 CROSS REFERENCES

Maximum Flow. Maximum Matching. Online Graph problems (Coloring, Matching). Online Matching.

9 RECOMMENDED READING

[1] J. Díaz, M. Serna, P.G. Spirakis, and J. Torán *Paradigms for fast parallel approximation.* Cambridge International Series on Parallel Computation, 8. Cambridge University Press, Cambridge, 1997.

[2] S. Even. *Graph Algorithms.* Computer Science Press, Potomac, Md, 1979.

[3] L.M. Goldschlager, R.A. Shaw, and J. Staples. The maximum flow problem is log-space complete for P. *Theoretical Computer Science*, 21:105–111, 1982.

[4] D.B. Johnson and S.M. Venkatesan. Parallel algorithms for minimum cuts and maximum flows in planar networks. *Journal of the ACM*, 34:950–967, 1987.

[5] R.M. Karp, E. Upfal, and A. Wigderson. Constructing a perfect matching is in Random NC. *Combinatorica*, 6:35–48, 1986.

[6] B. Korte and R. Schrader. On the existence of fast approximation schemes. *Nonlinear Programming*, 4:415–437, 1980.

[7] E.L. Lawler. *Combinatorial Optimization: Networks and Matroids.* Holt, Rinehart and Winston, NY, 1976.

[8] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, Mass., 1994.

[9] J.G. Peters and L. Rudolph. Parallel aproximation schemes for subset sum and knapsack problems. *Acta Informatica*, 24:417–432, 1987.

[10] P. Spirakis. PRAM models and fundamental parallel algorithm techniques: Part II. In A. Gibbons and P. Spirakis, editors, *Lectures on Parallel Computation*, pages 41–66. Cambrige University Press, 1993.