A fast parallel algorithm for minimum-cost small integral flows

Andrzej Lingas, Lund University

and

Mia Persson, Malmö University

A parallel random-access machine (PRAM)

- PRAM is a shared-memory abstract machine.
- It consists of a number of processors, each of which has its own local memory and can execute its own local program, and all of which communicate by exchanging data through a shared memory unit.
- All the processors operate synchronously under the control of a common clock.

PRAM variants

Depending on how read/write conflicts in accessing the same shared memory location simultaneously are resolved, one distinguish the followings variants of PRAM:

1. Exclusive read exclusive write (EREW) PRAM – every memory cell can be read or written to by only one processor at a time
2. Concurrent read exclusive write (CREW) PRAM – multiple processors can read a memory cell but only one can write at a time
3. Concurrent read concurrent write (CRCW) PRAM – multiple processors can read and write.

Strenghts and weaknesses of PRAM

- In the same way that the RAM is used by sequential-algorithm designers to model algorithmic performance (such as time complexity), the PRAM is used by parallel-algorithm designers to model parallel algorithmic performance (such as time complexity, where the number of processors assumed is typically also stated).
- Similar to the way in which the RAM model neglects practical issues, such as access time to cache memory versus main memory, the PRAM model neglects such issues as synchronization and communication, but provides any (problem-size-dependent) number of processors.
- Algorithm cost, for instance, is estimated using two parameters O(time) and O(time*processor-number).

NC parallel complexity class

- The class NC (for "Nick's Class") is the set of decision problems decidable in polylogarithmic time on a PRAM with a polynomial number of processors.
- In other words, a problem is in NC if there exist constants c and k such that it can be solved in time $O(log^c n)$ using PRAM with $O(n^k)$ parallel processors.
- Stephen Cook coined the name "Nick's class" after Nick Pippenger who had done extensive research on circuits with polylogarithmic depth and polynomial size.

NC^i parallel complexity classes

NC^i is the class of decision problems solvable by uniform Boolean circuits with a polynomial number of gates of at most two inputs and depth $O(log^i n)$, i.e.,

directed acyclic networks with terminals corresponding to n input variables, and non-terminal AND, OR, NEGATION nodes that for a given input size n can be generated by a $O(\log n)$-space algorithm.

RNC and RNC^i parallel complexity classes

RNC is defined analogously as NC by using a randomized PRAM instead of deterministic PRAM. A randomized PRAM has access to random bits and the probability that its output is not correct is very small (Monte Carlo model).

Similarly, RNC^i classes are defined analogously to the NC^i classes by equiping Boolean circuits additionally with random bit gates and allowing outputs with bounded error probability.

P-completeness

A decision problem is P-complete (complete for the complexity class PTIME) if it is in PTIME and that every problem in PTIME can be reduced to it by using an NC-reduction, i.e., the reduction can be performed by a PRAM in polylogarithmic time using polynomial number of processors.

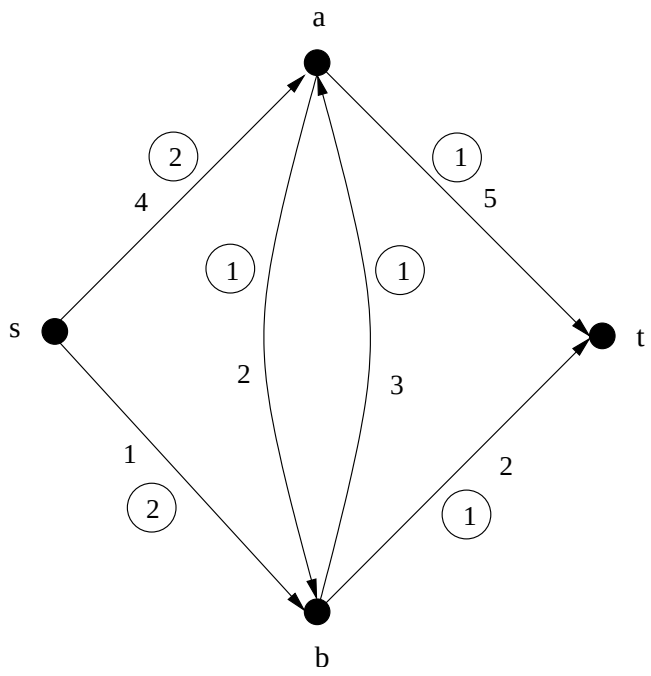
Note that if PTIME$\neq$ NC then all P-complete problems lie outside NC.

The minimum-cost flow problem

The minimum-cost flow problem is a well known important generalization of the maximum flow problem.

Definition. Let $G = (V, E)$ be a directed graph with a cost associated with each edge. For a flow f in G, the cost of f is $\sum_{e \in E} f(e)cost(e)$. The objective is to compute a maximum flow of minimum cost.

An example of the minimum-cost flow problem



A weighted flow network illustrating minimum-cost flow. The circled numbers are capacities and the encircled numbers are costs. The optimal $s - t$ flow is 2 and has cost 12.

Maximum and minimum-cost network flows

- Well-known problems in algorithms and optimization with plenty of important applications.
- Both problems are P-complete.
- In the proof of P-completeness one uses edge capacities of exponential size.
- The existing parallell algorithms for the general maximum and minimum-cost network flow problems require $\Omega(n^{\alpha})$ time, where α is a positive constant.

Maximum and minimum-cost network flows

- If edge capacities and flow supply are polynomially bounded then both problems admit RNC algorithms (more precisely, an RNC^2 algorithm for maximum integer flow and an RNC^3 for minimum-cost integer flow problem).
- At the heart of the aforementioned RNC solutions is the randomized method of detecting a perfect matching by randomly testing Edmonds' multi-variable polynomials for non-identity with zero.

Maximum and minimum-cost network flows

- When the flow supply is poly-logarithmic, then an NC implementation of the basic phase of Ford-Fulkerson yields an NC algorithm for maximum integer flow.
- This one can be extended to an NC algorithm for minimum-cost integer flow (if flow supply is logarithmic then we obtain an NC^2 algorithm for maximum integer flow and an NC^3 algorithm for minimum-cost integer flow).

Our results

- We present a new approach to the minimum-cost integral flow problem for a small value k of the flow. It relies on a fast randomized parallel method for a parameterized disjoint path problem.
- In our approach, we associate a simple polynomial over a finite field with the corresponding problem of the existence of k mutually vertex disjoint paths of bounded total length, connecting two sets of k terminals in a directed graph. Now, by using the idea of monomial cancellation, the latter problem reduces to testing the polynomial over a finite field of characteristic two for non-identity with zero.

Our results

- In effect, we infer that a minimum-cost flow of value k in a network with n vertices, a sink and a source, integral edge capacities and positive integral edge costs polynomially bounded in n can be found by a randomized PRAM, with errors of exponentially small probability in n, running in $O(k \log(kn) + \log^2(kn))$ time and using $2^k (kn)^{O(1)}$ processors.
- Thus, for flows of value $O(\log n)$, we obtain an RNC^2 algorithm for minimum-cost flow problem.

Connecting vertex-disjoint paths

Definition 1. Let $L = (V, E)$ be a network in a form of a directed graph with n vertices, among them a distinguished set $X = \{x_1, ..., x_k\}$ of k source vertices and a disjoint distinguished set $Y = \{y_1, ..., y_k\}$ of k sink vertices.

Definition 2. A *walk* in L is a sequence of vertices $v_1, v_2, ..., v_l$ of L such that for $j = 1, ..., l-1$, $(v_j, v_{j+1}) \in E$, v_1 is in X, $v_2, ..., v_{l-1}$ are in $V \setminus (X \cup Y)$, v_l is in Y. The length of the walk is $l - 1$.

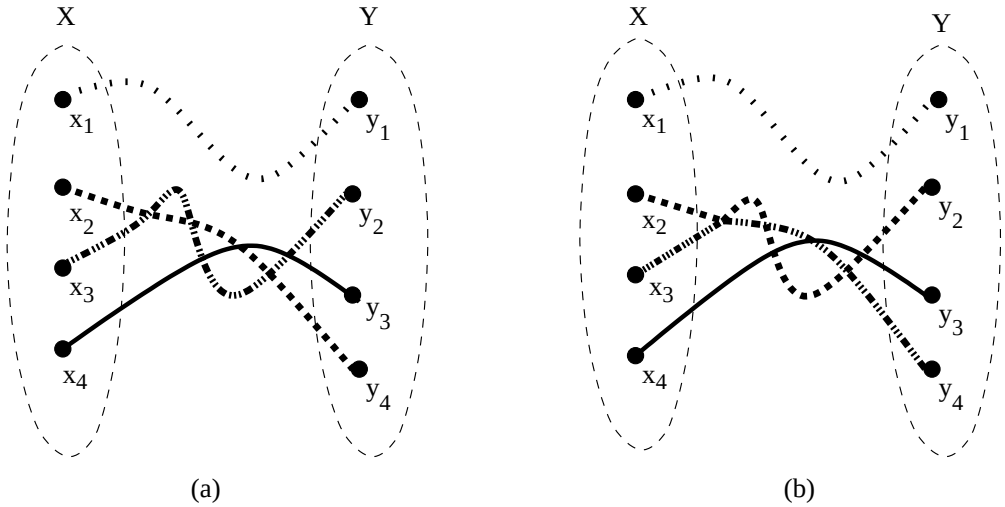
Definition 3. A *proper set S of walks* in L is a set W of k walks of total length $\leq k(n-1)$, each with a distinct start vertex in X and a distinct end vertex in Y.

Connecting vertex-disjoint paths

Definition 4. A *signature* of a proper set S of walks is the pair (i, j) that is smallest in lexicographic order such that the two walks that start at x_i and x_j, respectively, intersect and their first intersection vertex is also the first intersection vertex of the walk starting from x_i with any walk in S.

Connecting vertex-disjoint paths

- Note that walks in S are pairwise vertex disjoint iff the signature of S is not defined.
- We define the transformation ϕ on S as follows. If S has the signature (i, j) then ϕ switches the suffix of the walk starting at x_i with that of the walk starting at x_j at the first intersection vertex of these two walks (see figure below).



Connecting vertex-disjoint paths

- For the network L and $l \in [k(n-1)]$, let $F_{L,l}$ be the family of all proper sets of k walks of total length $\leq l$ in L.
- Assign a distinct variable x_e to each edge e in L, and for a walk W, let M_W be the monomial where x_e has multiplicity equal to the number of occurrences of e in W.
- Next, let $Q_{L,l}$ denote the polynomial $\sum_{S \in F_{L,l}} \prod_{W \in S} M_W$.

Lemma 1. For the network L and $l \in [k(n-1)]$, there is a proper set of k mutually vertex-disjoint walks of total length $\leq l$ in L iff $Q_{L,l}$ is not identical to zero over a field of characteristic two.

Proof idea. Partition $F_{L,l}$ into the two families $F^1_{L,l}$ (of sets S of walks such that $\phi(S) = S$) and $F^2_{L,l}$ (of sets S of walks such that $\phi(S) \neq S$).

Now, polynomial $\sum_{S \in F^2_{L,l}} \prod_{W \in S} M_W$ is identical to zero over a field of characteristic two.

On the other hand, since each set S of walks in $F^1_{L,l}$ consist of mutually vertex-disjoint walks, the monomials $\prod_{W \in S} M_W$ in the polynomial $\sum_{S \in F^1_{L,l}} \prod_{W \in S} M_W$ are in one-to-one correspondence with S and thus unique if the walks are simple paths. It is sufficient to observe that mutually vertex-disjoint walks can always be pruned to corresponding mutually vertex-disjoint simple paths.

Testing a polynomial for non-identity with 0

Lemma 2. [DeMillo and Lipton, Schwartz, and Zippel.] Let $Q(x_1, x_2, \ldots, x_m)$ be a nonzero polynomial of degree d over a field of size r. Then, for $f_1, f_2, \ldots, f_m$ chosen indepentently and uniformly at random form the field, the probability that $Q(f_1, f_2, \ldots, f_m)$ is not equal to zero is at least $1 - \frac{d}{r}$.

Since $Q_{L,l}$ is of degree $l \leq k(n-1) \leq n^2$, Lemma 2 can be used with a field $F_{2^{O(\log n)}}$ of characteristic two to obtain a randomized test of $Q_{L,l}$ for not being identical to zero with one side errors (for sufficient large c, we obtain one side errors not larger than a constant smaller than 1 with high probability).

By performing $O(n)$ such independent tests in parallel, the probability of one side errors can be decreased to exponentially small in n.

Parallel evaluation of the polynomial $Q_{L,l}$

By partially parallelizing a straightforward sequential evaluation of $Q_{L,l}$, we obtain:

Lemma 3. $Q_{L,l}$ can be evaluated for a given assignment of values over a field $F_{2^{O(\log n)}}$ of characteristic two in $O(k \log n + \log^2 n)$ time by a CREW PRAM using $O(kn^4 + 2^k k^3 n^2)$ processors.

Connecting vertex-disjoint paths

By Lemma 3, the series of the tests can be performed in $O(k \log n + \log^2 n)$ time by a PRAM using $O(kn^4 + 2^k k^3 n^3)$ processors. Hence, by Lemma 1, we obtain:

Theorem 1. The problem of whether or not there is a set of k mutually vertex-disjoint simple directed paths of total length l connecting X with Y in the network L can be decided by a randomized CREW PRAM, with one-sided errors of exponentially small probability in n, running in $O(k \log n + \log^2 n)$ time and using $O(kn^4 + 2^k k^3 n^3)$ processors.

Vertex-disjoint paths of bounded cost

Suppose that there is given a positive integer C and a cost function c assigning to each of the m edges e in the network L a cost $c(e) \in [C]$.

We can solve the problem of deciding if there is a proper set of mutually vertex-disjoint walks in L of total cost U in similar manner by a straightforward reduction to that with total length constraint l. Roughtly, we replace each edge e with a directed path of length $c(e)$ introducing $c(e) - 1$ auxiliary vertices. Hence, we otain:

Theorem 2. The minimum cost of a set of k mutually vertex-disjoint simple directed paths connecting X with Y in the network L with edge costs in $[C]$ can be computed by a randomized CREW PRAM, with errors of exponentially small probability in n, running in $O(k \log(Cn) + \log^2(Cn))$ time and using $O(kC^5n^{10} + 2^k k^3 C^3 n^7)$ processors.

Finding vertex-disjoint paths of bounded cost

Lemma 4.(The isolation lemma) Let F be a family of subsets of a set with q elements and let r be a non-negative integer. Suppose that each element s of the set is independently assigned a weight $w(s)$ uniformly at random from $[r]$, and the weight of a subset S in F is defined as $w(S) = \sum_{x \in S} w(x)$. Then, the probability that there is a unique set in F of minimum weight is at least $1 - \frac{q}{r}$.

Corollary 1. For each of the m edges e in the network L, modify its cost $c(e)$ to $c'(e) = c(e)rm + w(e)$, where the weight $w(e)$ is drawn uniformly at random from $[r]$. Then, the probability that there is a unique minimum-cost set of mutually vertex-disjoint paths connecting X with Y in the edge weighted network L is at least $1 - \frac{m}{r}$.

Finding vertex-disjoint paths of bounded cost

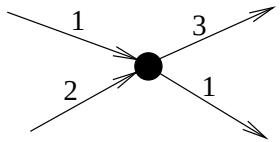
By combining the corollary from the isolation lemma with the polynomial-testing method for the decision version of minimum-cost vertex-disjoint connecting path problem (Theorem 2), we can test each edge of the network for the membership in the optimal solution independently in parallel.

We obtain the following solution to the corresponding search problem.

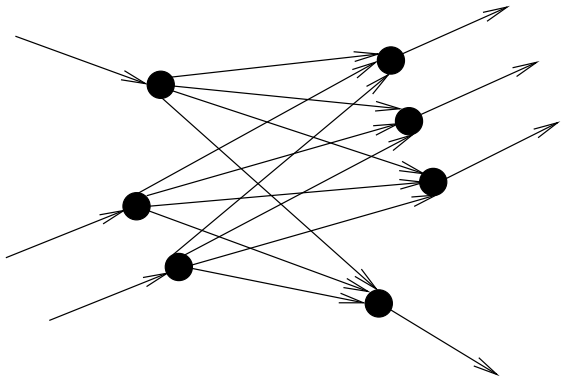
Theorem 3. There is a randomized PRAM returning almost certainly (i.e., with probability at least $1 - \frac{1}{n^\alpha}$, where $\alpha \geq 1$) a minimum-cost set of k mutually vertex-disjoint paths connecting X with Y in the network L with the original edge costs in $[C]$ (iff such a set exists) in $O(k \log(Cn) + \log^2(Cn))$ time using $2^k (kCn)^{O(1)}$ processors.

Parallel minimum-cost integral flow

The minimum-cost integral flow problem admits a straightforward reduction to the corresponding minimum-cost disjoint connecting path problem. Roughly, we replace each edge with the number of parallel edges equal to its capacity and each vertex with an appropriate complete bipartite directed graph (see figure below).



(a)



(b)

Theorem 4. The minimum cost of a flow of value k in a network with n vertices, a sink and a source, integral edge capacities and positive integral edge costs in $[C]$ can be found by a randomized PRAM, with errors of exponentially small probability in n, running in $O(k \log(Ckn) + \log^2(Ckn)$ time and using $O(k^{11}C^5n^{20} + 2^k k^{10}C^3n^{14})$ processors.

Theorem 5. There is a randomized PRAM algorithm returning almost certainly a minimum-cost flow of value k (iff a flow of value k exists) in a network with n vertices, a sink and a source, integral edge capacities and edge costs in $[n^{O(1)}]$, in $O(k \log(kn) + \log^2(kn))$ time using $2^k(kn)^{O(1)}$ processors.

Corollary 2. The problem of finding a minimum-cost flow of value $O(\log n)$ in a network with n vertices, a sink and a source, and integral edge capacities bounded polynomially in n admits an RNC^2 algorithm.

Thank you!