A Fast Parallel Algorithm
for Minimum-Cost Small Integral Flows

Andrzej Lingas[1] and Mia Persson[2]

[1] Department of Computer Science, Lund University, 22100 Lund, Sweden
Andrzej.Lingas@cs.lth.se
[2] Department of Computer Science, Malmö University, 205 06 Malmö, Sweden
mia.persson@mah.se

Abstract. We present a new approach to the minimum-cost integral flow problem for small values of the flow. It reduces the problem to the tests of simple multi-variable polynomials over a finite field of characteristic two for non-identity with zero. In effect, we show 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, in particular, for the minimum-cost flow of value $O(\log n)$, we obtain an RNC^2 algorithm.

1 Introduction

The maximum network flow problem is a well known fundamental problem in algorithms and optimization with plenty of important applications [1, 7, 8, 16]. It is known to be P-complete even in its integral version provided that the edge capacities are exponentially large in the size of the network [13]. The minimum-cost flow problem is a well known important generalization of the maximum flow problem [1, 8, 10, 16]. The objective is to compute a maximum flow of minimum cost in a directed graph where each edge is assigned a cost. For a flow f in a directed graph (V, E), the cost of f is simply $\sum_{e \in E} f(e)cost(e)$.

The prospects for designing a fast and processor efficient parallel algorithm, in particular an NC algorithm [19], for maximum integral flow or minimum-cost integral flow are small. The fastest known parallel implementations of general maximum flow and/or minimum-cost flow algorithms achieve solely a moderate speed up and still run in $\Omega(n^\alpha)$ time, where α is a positive constant, see [2, 12].

The situation changes when the edge capacities or the supply of flow as well as edge costs are substantially bounded. For example, if the edge capacities and edge costs are bounded by a polynomials in n, both problems admit RNC algorithms. Then, the maximum integer flow problem admits even an RNC^2 algorithm [17, 18, 20] while the minimum-cost integer flow problem admits an RNC^3 algorithm [18]. 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 [6, 14, 17, 20].

When the flow supply is relatively small, e.g., logarithmic in the size of the network or a poly-logarithmic one, then just an NC implementation of the basic phase in the standard Ford-Fulkerson method [1, 7–9, 16] yields an NC algorithm (NC^3 when the supply is logarithmic) for maximum integer flow that can be extended to an NC algorithm for minimum-cost integer flow (when edge costs are polynomially bounded). The number of processors used corresponds to that required by a shortest path computation.

In this paper, we present a new approach to the minimum-cost integral flow problem for a small value k of the flow. We directly 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. 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. We combine the DeMillo-Lipton-Schwartz-Zippel lemma [5, 21] on probabilistic verification of polynomial identities with parallel dynamic programming to perform the test efficiently in parallel. Additionally, we use the isolation lemma to construct the minimum-cost flow [17, 20].

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, in particular, for the minimum-cost flow of value $O(\log n)$, we obtain an RNC^2 algorithm.

Related Work. For the RNC algorithms for the related problem of minimum-cost perfect matching see [14, 17, 20]. For the comparison of time and substantial processor complexities of prior RNC algorithms for the minimum-cost flow see page 7 in [18]. The fastest of the reported algorithms is not an RNC^2 one even when the flow supply and thus the edge capacities are logarithmic in the size of the network. The idea of associating a polynomial over a finite field to the sought structure has been already used by Edmonds to detect matching [6] and then in several papers presenting RNC algorithms for perfect matching construction [14, 17, 20]. It appears in several recent papers that also exploit the idea of monomial cancellation [3, 4, 15, 22].

Organization. In the next section, we comment briefly on the basic notation and the model of parallel computation used in the paper. In Section 3, we derive our fast randomized parallel method for detecting the existence of k mutually vertex disjoint paths of bounded total length connecting two sets of k terminals in a directed graph. In Section 4, we generalize the method to include edge costs which enables us to replace the total length bound with the total cost one. In section 5, we show a straightforward reduction of the minimum-cost integer flow problem parametrized by the flow value to the corresponding disjoint paths problem which enables us to derive our main result on detecting minimum-cost small flows in parallel.

2   Terminology

For a natural number n, we let $[n]$ denote the set of natural numbers in the interval $[1, n]$. The cardinality of a set A will be denoted by $|A|$.

We assume the standard definitions of *flow* and *flow value* in a network (directed graph) with integral edge capacities, a distinguished source vertex s and a distinguished sink vertex t (e.g., see [7]) .

For the definitions of parallel random access machines (PRAM), the classes NC and RNC and the corresponding notions of NC and RNC algorithms, the reader is referred to [19].

The *characteristic* of a ring or a field is the minimum number of 1 in a sum that yields 0. A finite field with q elements is often denoted by F_q.

3   Connecting Vertex-Disjoint Paths

It is well known that the maximum integral network flow problem with bounded edge capacities corresponds to a disjoint path problem (cf. [7]). In Section 5, we provide an efficient parallel reduction of the minimum-cost integral flow problem parametrized by the flow value to a parametrized disjoint path problem. This section is devoted to a derivation of a fast randomized parallel method for the decision version of the parametrized path problem.

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.

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 $l-1$. In other words, a walk is just a (not necessarily simple) path starting from a vertex in X, having intermediate vertices in $V \setminus (X \cup Y)$, and ending at a vertex in Y.

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.

A *signature* of S 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.

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 Fig. 1. Otherwise, if the signature of S is not defined then ϕ is an identity on S.

Observe that if the signature of S is defined then $\phi(S)$ has the same signature as S and $\phi(S) \neq S$. The first observation is immediate. To show the second one it is sufficient to note that $\phi(S) = S$ holds iff ϕ transforms the two walks which yield the signature of S onto themselves. The latter is however impossible since they have different start vertices and different end vertices. Note also that the walks in $\phi(S)$ have the same total length as those in S.
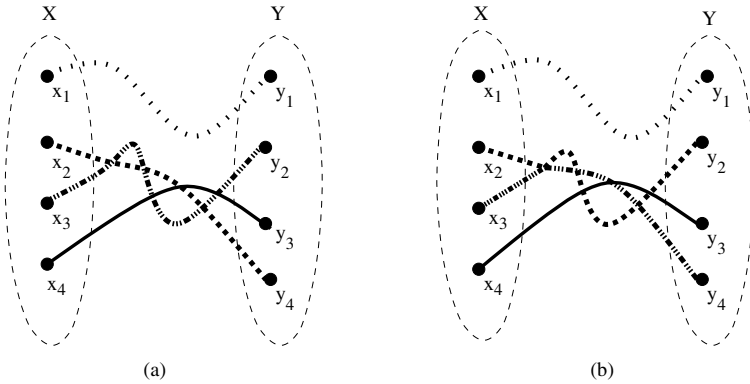
Fig. 1. An example of a proper set S of walks and the companion proper set $\phi(S)$ of walks

It follows that ϕ is an involution on sets of proper walks of total length l, i.e., $\phi(\phi(S)) = S$ holds for any proper set S of walks of total length l.

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 l in L. Assign a distinct variable x_e to each edge e in L. For a walk $W \in F_{L,l}$, 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 l in L iff $Q_{L,l}$ is not identical to zero over a field of characteristic two.*

Proof. $F_{L,l}$ can be partitioned into the family $F_{L,l}^1$ of sets S of walks such that $\phi(S) = S$ and the family $F_{L,l}^2$ of sets S of walks such that $\phi(S) \neq S$. The polynomial $\sum_{S \in F_{L,l}^2} \prod_{W \in S} M_W$ is identical to zero over a field of characteristic two since for each $S \in F_L^2$ the monomials $\prod_{W \in S} M_W$ and $\prod_{W \in \phi(S)} M_W$ contain equal multiplicities of the same variables and $\phi(\phi(S)) = S$ so S and $\phi(S)$ can be paired. On the other hand, since each set S of walks in $F_{L,l}^1$ consist of mutually vertex-disjoint walks, the monomials in the polynomial $\sum_{S \in F_{L,l}^1} \prod_{W \in S} M_W$ are in one-to one correspondence with S and therefore are unique.
□

To warm up, we prove the following lemma on sequential evaluation of Q_L.

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

Proof. For $B \subset Y$, $l \in [(n-1)|B|]$, we consider the family $W_l(B)$ of all sets S consisting of $|B|$ walks connecting $|B|$ distinct sources in $\{x_1, ..., x_{|B|}\}$ with the $|B|$ distinct sinks in B so that the total length of the walks is exactly l. Next, we define the polynomial $Q_l(B)$ as $\sum_{S \in W_l(B)} \prod_{W \in S} M_W$. Note that $Q_{L,l} = Q_l(Y)$.

On the other hand, for $l \in [k(n-1)]$, $x \in X$ and $z \in V \setminus X$, we consider the set $W_l(x, z)$ of walks of length l in L that start at x and end at z. Let $Q_l(x, z)$ be the polynomial $\sum_{W \in W_l(x,z)} M_W$.

We have the following recurrence for a nonempty subset B of Y and $l \in [|B|(n-1)]$:

$$Q_l(B) = \sum_{y \in B} \sum_{q \in [|B|(n-1)-|B|+1]} Q_{l-q}(B \setminus \{y\})Q_q(x_{|B|}, y).$$

Next, we have also the following recurrence for $x \in X$, $z \in V \setminus X$, and $q \in [k(n-1)]$:

$$Q_q(x, z) = \sum_{u \in V \setminus (X \cup Y) \& (u,z) \in E} Q_{q-1}(x, u)x_{(u,z)}.$$

We have also $Q_1(x, z) = x_{(x,z)}$ if $(x, z) \in E$, and otherwise $Q_1(x, z) = 0$. Consequently, we can evaluate all the polynomials $Q_q(x, z)$ by the second recurrence in $O(k^2 n^3)$ time.

Now, by using the first recurrence and setting $Q_l(\emptyset)$ to 1 in the field, we can evaluate all the polynomials $Q_l(B)$ in the increasing order of the cardinalities of B in $O(2^k k^3 n^2)$ time.

□

We can partially parallelize the sequential evaluation of $Q_{L,l}$ in order to obtain the following lemma.

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.*

Proof. We generalize the definition of the set $W_q(x, z)$ and the corresponding polynomial $Q_q(x, z)$ to include arbitrary start vertex $x \in V \setminus Y$, requiring $z \in V \setminus X$ as previously. Then, we can evaluate $Q_q(x, z) = \sum_{W \in W_l(x,z)} M_W$ for $x \in V \setminus Y$ and $z \in V \setminus X$, for $q \in [k(n-1)]$ by the following standard doubling recurrence for $q \geq 2$:

$$Q_q(x, z) = \sum_{y \in V \setminus (X \cup Y)} Q_{\lceil q/2 \rceil}(x, y)Q_{\lfloor q/2 \rfloor}(y, z)$$

At the bottom of the recursion, we have $Q_1(x, z) = x_{(x,z)}$ if $(x, z) \in E$, otherwise $Q_1(x, z) = 0$. It follows that all $Q_q(x, z)$ for $x \in V \setminus Y$, $z \in V \setminus X$, and $q \in [k(n-1)]$ can be evaluated in a bottom-up manner in $O(\log^2 n)$ time by a CREW PRAM using $O(kn^4)$ processors.

Recall the first recurrence from the proof of Lemma 2. When the polynomials $Q_q(x_i, y_j)$ for $x_i \in X$ and $y_i \in Y$ are evaluated, we can evaluate in turn the polynomials $Q_l(B)$, where $B \subset Y$, $l \in [|B|(n-1)]$ in k phases in the increasing order of the cardinalities of B by this recurrence. It can be done in $O(k(\log k + \log n)) = O(k \log n)$ time by a CREW PRAM using $2^k k^3 n^2$ processors.

By $Q_{L,l} = Q_l(Y)$, $k \leq n$, we conclude that $Q_{L,l}$ can be evaluated in $O(k \log n + \log^2 n)$ time by a CREW PRAM using $O(kn^4 + 2^k k^3 n^2)$ processors.

$\square$

The following lemma on polynomial identities verification has been shown independently by DeMillo and Lipton, Schwartz, and Zippel.

Lemma 4. *[5, 21] Let $Q(x_1, x_2, ..., x_m)$ be a nonzero polynomial of degree d over a field of size r. Then, for f_1, f_2, ...,f_m chosen independently and uniformly at random from the field, the probability that $Q(f_1, f_2, ..., f_m)$ is not equal to zero is at least $1 - \frac{d}{r}$.*

Note that the polynomial $Q_{L,l}$ is of degree l not larger than $k(n-1) \leq n^2$. We can use Lemma 4 with a field $F_{2^{c \log n}}$ of characteristic two to obtain a randomized test of the polynomial $Q_{L,l}$ for not being identical to zero with one side errors. For sufficiently large constant c, the one side errors are of probability not larger than a constant smaller than 1. By performing $O(n)$ such independent tests, the probability of one side errors can be decreased to exponentially small in n one.

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. By Lemma 1, these tests verify if there is a proper set of mutually vertex-disjoint walks of total length l in the network L. The latter in turn is equivalent to the existence of k mutually vertex-disjoint paths of total length l connecting X with Y in L by the definition of a proper set of walks in L. Hence, observing that each walk can be trivially pruned to a simple directed path with the same endpoints, we obtain our main result.

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.*

4   Vertex-Disjoint Connecting Paths of Bounded Cost

In this section, we shall consider a more general situation where there are a positive integer C and a cost function c assigning to each of the m edges e in the network L a cost $c(e) \in [C]$. The cost of a walk or a path is simply the sum of the costs of the edges forming it (the cost of an edge is counted the number of times it appears on the walk or path). We would like to detect a proper set of k walks in L that achieves the minimum cost.

For this reason, we consider the following generalization of the polynomial $Q_{L,l}$. For $U \in [mC]$, let $H_{L,U}$ be the set of all proper sets of walks in the edge-costed network L that have total cost equal to U. Next, for a walk W in L, as previously, let M_W be the monomial which is the product of x_e over the occurrences of edges e on W. The polynomial $CQ_{L,U}$ is defined by $\sum_{S \in H_{L,U}} \prod_{W \in S} M_W$.

By using the proof method of Lemma 1, we obtain the following counterpart of this lemma for $CQ_{L,U}$.

Lemma 5. *For the edge-costed network L, there is a proper set of k mutually vertex-disjoint walks of total cost U in L iff $CQ_{L,U}$ is not identical to zero over a field of characteristic two.*

Next, we obtain the following counterpart of Lemma 3 for $CQ_{L,U}$.

Lemma 6. *$CQ_{L,U}$ can be evaluated for a given assignment f of values over a field $F_{2^{O(\log n)}}$ of characteristic two in $O(k \log(Cn) + \log^2(Cn))$ time by a PRAM using $O(kC^4 n^8 + 2^k k^3 C^2 n^4)$ processors.*

Proof. The proof reduces to that of Lemma 3. We replace each directed edge e of cost $c(e) \in [C]$ in the network L by a directed path of length $c(e)$ introducing $c(1) - 1$ additional vertices. With each edge on such a path, we associate a variable. We assign $f(x_e)$ to the variable associated with the first edge on the path replacing e, and just 1 of the field to the variables associated with the remaining edges on the path.

The resulting network L' is of size $O(Cn^2)$. Let $H_{L',U}$ be the family of all proper sets of k walks of total cost U in the network L'. We can evaluate the polynomial $Q_{L',U} = \sum_{S \in H_{L',U}} \prod_{W \in S} M_W$ in parallel analogously as $Q_{L,l}$ in the proof of Lemma 3. It remains to observe that the value of $CQ_{L,U}$ under the assignment f is equal to that of $Q_{L',U}$ under the aforementioned assignment. □

Now, we are ready to derive our main result in this section.

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 $O(kC^5 n^{10} + 2^k k^3 C^3 n^7)$ processors.*

Proof. The minimum cost of the sought set of vertex-disjoint paths is in $[Cn^2]$. Hence, by Lemma 5, it is sufficient to test the polynomials $CQ_{L,U}$ for non-identity with zero for all $U \in [Cn^2]$ in parallel. By applying Lemmata 4 and 6 in a manner analogous to the proof of Theorem 1, we conclude that it can be done by a randomized CREW PRAM, with one-sided errors of exponentially small in n probability, running in $O(k \log(Cn) + \log^2(Cn))$ time and using $O(Cn^2 \times (kC^4 n^8 + 2^k k^3 C^2 n^5))$ processors. □

5   Finding Vertex-Disjoint Connecting Paths

A straightforward approach of extending our randomized parallel method for deciding if there is a proper set of k mutually vertex-disjoint walks (of a given total cost) between two sets of vertices of cardinality k to include the finding variant could be roughly as follows. In parallel, for each k-tuple of respective neighbors of the k start vertices in X, replace the set of start vertices by the k-tuple and apply our method recursively to the resulting network. If the test is positive, the first edges on the walks are known, and we can iterate the method. The problem with this approach is that its recursive depth is proportional to the maximum length of a walk in the resulting set of mutually vertex-disjoint walks between X and Y.

Also, it is not clear how one could implement a straightforward divide-and-conquer approach of guessing intermediate vertices in order to find a set of k mutually-vertex disjoint walks of a given cost efficiently in parallel.

We need more advanced methods to obtain a very fast parallelization of the finding variant. We shall modify the edge cost in the network L in order to use the so called *isolation lemma* in a manner analogous to the RNC method of finding a perfect matching given in [20].

Lemma 7. *(The isolation lemma [20]). 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 $1 - \frac{m}{r}$.*

Proof. To use the isolation lemma, let the underlying set to consist of all edges in the network L. Next, note that a set of mutually vertex-disjoint paths connecting X with Y achieving a minimum cost consists of simple paths and thus it can be identified with the set of edges on the paths. Let P be the family of all sets of mutually vertex-disjoint simple paths connecting X with Y in the network L. By the setting of new costs $c'(e)$, solely those sets in P that achieved the minimum cost, say D, under the original costs $c(e)$ can achieve a minimum cost under the new costs $c'(e)$. So, we can set F to the aforementioned sub-family of P, and define the weight of a set of k paths in F as the sum of the weights $w(e)$ of the edges e on the paths in this set in order to use the isolation lemma. By the isolation lemma, there is a unique set S in F that achieves the minimum weight $w(S)$ with the probability at least $1 - \frac{m}{r}$. The corollary follows since each set S in F has the cost $c'(S)$ equal to $D + w(S)$.   □

Throughout the rest of this section, we shall assume that each of the m edges e in the network L is assigned the cost $c'(e)$ as in Corollary 1 and that $r \in [n^{O(1)}]$.

Suppose that we know the minimum cost of a set of k mutually vertex-disjoint paths connecting X with Y in the network L with the edge costs indicated, and such a minimum-cost set is unique. Then, it is sufficient to show that we can test quickly in parallel if the network L with an arbitrary edge removed still contains a set of k mutually vertex-disjoint paths connecting X with Y that achieves the minimum cost. By performing the test for each edge of L in parallel, we can determine the set of edges forming the unique minimum-cost set of k mutually vertex-disjoint paths connecting X with Y.

To carry out these tests, we need to generalize the polynomial $CQ_{L,U}$ to a polynomial $CP_{L,e,U}$, where e is an edge in L and U is a cost constraint from $[mr(mC+1)] = [Cn^{O(1)}]$. Let $H_{L,e,U}$ be the family of all proper sets of k walks in the network L with the edge e removed that have total cost equal to U. (In the total cost of a set of walks, we count the cost of an edge the number of times equal to the sum of the multiplicities of the edge in the walks.)

As in the definition of $Q_{L,l}$ assign a distinct variable x_e to each edge e in L, and for a walk $W \in H_{L,e,W}$, let M_W be the monomial, where x_e has multiplicity equal to the number of occurrences of e in W. The polynomial $CP_{L,e,U}$ is defined by $\sum_{S \in H_{L,e,W}} \prod_{W \in S} M_W$.

By using the proof method of Lemma 1, we obtain the following counterpart of this lemma for $CP_{L,e,U}$.

Lemma 8. *For the edge-costed network L with m edges, edge e, and $U \in [Cn^{O(1)}]$, there is a proper set of k mutually vertex-disjoint walks of total cost U in the network L with the edge e removed iff $CP_{L,e,U}$ is not identical to zero over a field of characteristic two.*

Next, we obtain the counterpart of Lemma 3 for $CP_{L,e,U}$ following the proof of Lemma 6.

Lemma 9. *$CP_{L,e,U}$ can be evaluated for a given assignment of values over a field $F_{2^{O(\log n)}}$ of characteristic two in $O(k \log(Cn) + \log^2(Cn))$ time by a PRAM using $2^k (kCn)^{O(1)}$ processors.*

Now, we are ready to derive our main result in this section.

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.*

Proof. We set r to, say, $n^2 m$, and specify the new edge costs $c'(e)$ in the network L drawing the weights $w(e)$ uniformly at random from $[r]$ as in Corollary 1. Next, for each $U \in [mr(mC+1)] = [Cn^{O(1)}]$, we proceed in parallel as follows. For each edge e of the network L, we test the polynomial $CP_{L,e,U}$ for the non-identity with zero by using Lemma 4 and Lemma 9 (we can perform a linear in n number of such tests in parallel in order to decrease the probability of the one-sided error to an exponentially small one). Next, we verify if the edges that passed the test

positively yield a set of k mutually vertex-disjoint paths connecting X with Y. For example, it can be done by checking for each endpoint of the edges outside $X \cup Y$ if it is shared by exactly two of the edges, and then computing and examining the transitive closure of the graph induced by the edges (see [19]). If so, we save the resulting set of paths of total (new) cost U. By Corollary 1, there is a $U \in [Cn^{O(1)}]$ for which the above procedure will find such a set of paths that achieves the minimum (original) cost with probability at least $1 - \frac{1}{n}$. $\square$

6   Minimum-Cost Logarithmic Integral Flow is in RNC^2

The following lemma is a straightforward generalization of a folklore reduction of maximum integral flow to a corresponding disjoint connecting path problem (for instance cf. [7]) to include minimum-cost integral flow. We shall call a flow proper, if it ships each flow unit along a simple path from the source to the sink.

Lemma 10. *The problem of whether or not there is a proper integral flow of value k and cost D from a distinguished source vertex s to a distinguished sink vertex t in a directed network with n vertices. integral edge capacities and edge costs in $[C]$ can be (many-one) reduced to that of whether or not there is set of k mutually vertex-disjoint simple directed paths of total cost D^*, where $\lfloor \frac{D^*}{kn} \rfloor = D$, connecting two distinguished sets of k vertices in a directed network on $O(kn^2)$ vertices in $O(\log k + \log n)$ time by a CREW PRAM using $O(kn^2 + k^2n)$ processors.*

Proof. Let $K = (V, E)$ be the directed network with integral edge capacities, edges costs in $[C]$ and the distinguished source vertex s and sink vertex t. Since we are interested in a flow of value k, we can assume w.l.o.g that all edge capacities do not exceed k.

We form a directed network K^* on the basis of the network K as follows.

Let $v \in V$. Next, let $E_{in}(v)$ be the set of edges in K incoming into v, and let $E_{out}(v)$ be the set of edges in K leaving v. For each $e \in E_{in}(v)$ and $i \in [capacity(e)]$, we create the vertex $v_{in}(e, i)$. Analogously, for each $e' \in E_{out}(v)$ and $i' \in [capacity(e')]$, we create the vertex $v_{out}(e', i')$. Furthermore, we direct an edge from each vertex $v_{in}(e, i)$ to each vertex $v_{out}(e', i')$. To each such an edge, we assign the cost 1. Also, for each edge $e = (v, w)$ of K, we direct an edge from $v_{out}(e, i)$ to $w_{in}(e, i)$ for $i \in [capacity(e)]$. To each such an edge, we assign the cost $c(e)kn$. See Fig. 2.   Let X' be the set of vertices of the form $s_{out}(...)$, and let Y' denote the set of vertices of the form $t_{in}(...)$. Create an additional set X of k vertices and from each vertex in X direct an edge to each vertex in X'. Symmetrically, create another additional set Y of k vertices and from each vertex in Y' direct an edge to each vertex in Y.

It is easy to observe that there is a proper integral flow of value k and cost D from s to t in the network K iff there is a set of k mutually vertex-disjoint simple paths of total cost D^* connecting X with Y in the network K^*, such that $\lfloor \frac{D^*}{kn} \rfloor = D$.
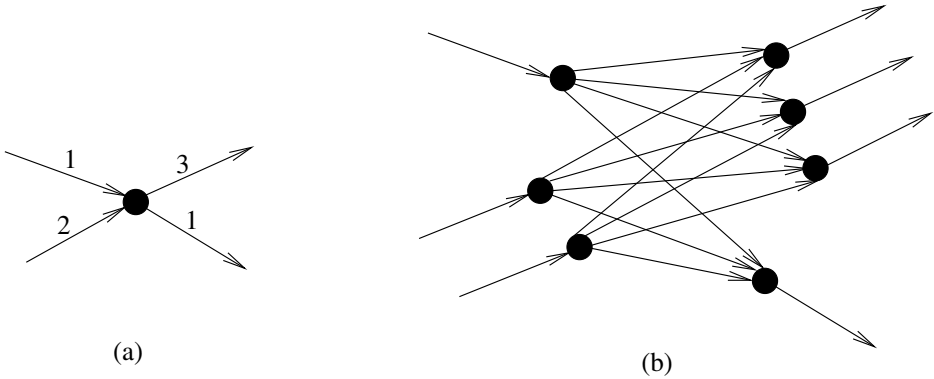
Fig. 2. An example of a vertex of the network K and the corresponding part of the network K^*

Now it is sufficient to note that the construction of K^*, X and Y on the basis of K easily implemented by a CREW PRAM in $O(\log k + \log n)$-time using $O(kn^2 + k^2 n)$ processors, where n is the number of vertices in K.   □

By combining Theorem 1 with Lemma 10, we obtain our first main result.

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 $O(k^{11} C^5 n^{20} + 2^k k^{10} C^3 n^{14})$ processors.*

By combining in turn Theorem 3 with the finding variant of Lemma 10 using exactly the same reduction, we obtain our second main result.

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.*

7   Final Remarks

We have resented a new approach to the minimum-cost integral flow problem. In particular, it yields an RNC^2 algorithm when the flow supply is (at most) logarithmic in the size of the network.

All our results can be extended to to include undirected networks by a straightforward reduction.

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms, and Applications. Prentice-Hall, Inc. (1993)
2. Anderson, R.J., Setubal, J.C.: On the Parallel Implementation of Goldberg's Maximum Flow Algorithm. In: Proc. Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA), pp. 168–177 (1992)
3. Björklund, A.: Determinant sums for undirected Hamiltonicity. In: Proc. 51th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 173–182 (2010)
4. Björklund, A., Husfeldt, T., Taslaman, N.: Shortest Cycle Through Specified Elements. In: Proc. Annual ACM-SIAM Symposium on Discrete Algorithms, SODA (2012)
5. DeMillo, R.A., Lipton, R.J.: A probabilistic remark on algebraic program testing. Information Processing Letters 7, 193–195 (1978)
6. Edmonds, J.: Systems of distinct representatives and linear algebra. J. Res. Nat. Bur. Standards Sect. 71B(4), 241–245 (1967)
7. Even, S.: Graph Algorithms. Computer Science Press Inc. (1979)
8. Edmonds, J., Karp, R.M.: Theoretical improvements in algorithmic efficiency for network flow problems. Journal of the ACM 19(2), 248–264 (1972)
9. Ford Jr., L.R., Fulkerson, D.R.: Maximal flow through a network. Can. J. Math. 8, 399–404
10. Fulkerson, D.R.: An out-of-Kilter method for minimal cost flow problems. SIAM J. Appl. Math. 9, 18–27
11. Galil, Z., Pan, V.: Improved processor bounds for algebraic and combinatorial problems in RNC. In: Proc. 26th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 490–495 (1985)
12. Goldberg, A.: Parallel Algorithms for Network Flow Problems. In: Reif, J.H. (ed.) Synthesis of Parallel Algorithms. Morgan-Kauffman (1993)
13. Goldschlager, L.M., Shaw, R.A., Staples, J.: The Maximum Flow Problem is Log Space Complete for P. Theoretical Computer Science 21(1), 105–111 (1982)
14. Karp, R., Upfal, E., Wigderson, A.: Constructing a perfect matching is in random NC. Combinatorica 6(1), 35–48 (1986)
15. Koutis, I.: Faster Algebraic Algorithms for Path and Packing Problems. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 575–586. Springer, Heidelberg (2008)
16. Lawler, E.L.: Combinatorial optimization: Networks and matroids. Holt, Rinehart and Winston, New York
17. Mulmuley, K., Vazirani, U.V., Vazirani, V.V.: Matching is as Easy as Matrix Inversion. Combinatorica 7(1), 105–113 (1987)
18. Orlin, J.B., Stein, C.: Parallel Algorithms for the Assignment and Minimum-Cost Flow Problems. Operations Research Letters 14, 181–186 (1993)
19. Reif, J.H.: Synthesis of Parallel Algorithms. Morgan-Kauffman (1993)
20. Vazirani, V.V.: Parallel Graph Matching. In: Reif, J.H. (ed.) Synthesis of Parallel Algorithms. Morgan-Kauffman (1993)
21. Schwartz, J.T.: Fast probabilistic algorithms for verification of polynomial identities. Journal of the ACM 27(4), 701–717 (1980)
22. Williams, R.: Finding paths of length k in $O^*(2^k)$. Information Processing Letters 109, 301–338 (2009)