

Distributed algorithm for approximating the maximum matching

A. Czygrinow

*Department of Mathematics and Statistics
Arizona State University
Tempe, AZ 85287-1804, USA
andrzej@math.la.asu.edu.*

M. Hańčkowiak

*Faculty of Mathematics and Computer Science
Adam Mickiewicz University
Poznań, Poland
mhanckow@main.amu.edu.pl.*

E. Szymańska

*Faculty of Mathematics and Computer Science
Adam Mickiewicz University
Poznań, Poland,
edka@main.amu.edu.pl.*

Abstract

We present a distributed algorithm that finds a matching M of size which is at least $2/3|M^*|$ where M^* is a maximum matching in a graph. The algorithm runs in $O(\log^6 n)$ steps.

Key words: distributed algorithm, maximum matching.

1 Introduction

We consider a distributed model of computations introduced by Linial in [Li92]. In this model a network is represented by an undirected graph where

¹ Research supported by KBN grant no. 7 T11C 032 20

each vertex represents a processor and each edge corresponds to a connection between processors in the network. In addition, each processor has a unique ID and knows the number of vertices in the graph. However, the global topology of the network is unknown. We assume full synchronization of the network: the computations are performed in steps. In a single step, each processor can send a message to all of its neighbors, can collect messages from its neighbors, and can perform some local computations.

The above model differs from classical model of parallel computations that uses shared memory. In the shared memory model processors can freely exchange information about current status of their computation using shared memory. In contrast, in the distributed model, vertices that are at "large" distance from each other in the graph cannot learn about each other in a reasonably short amount of time. This restriction poses new challenges in designing algorithms in the distributed model and many problems that admit simple and efficient solutions in PRAM model elude efficient algorithms in the distributed model. In fact, there are very few problems for which a poly-logarithmic distributed algorithm is known. One of such problems is the maximal matching problem, in which one searches for a matching M in a graph so that there is no matching M' that contains M and such that $|M| < |M'|$. In [HKP01], Hańcówkiak, Karoński, and Panconesi presented a distributed algorithm that finds a maximal matching in $O(\log^4 n)$ steps. In this paper, we will try to go one step further and find a matching of size that is closer to the size of the largest possible matching in the graph. Our strategy is first to apply the procedure from [HKP01] and find a maximal matching M and then to find a maximal independent set of paths of length three augmenting matching M . For each such path we delete from M an edge of the path that belongs to M and add to M the remaining two edges of the path. In this way, we obtain a new maximal matching M' such that there are no paths of length three augmenting M' . Therefore M' satisfies the assumption of the following theorem (see [FGHP93]).

Theorem 1 *Let M^* be a matching in a graph G that has the largest size. If there are no paths of length at most three augmenting a matching M then*

$$|M| \geq \frac{2}{3}|M^*|.$$

Note that our algorithm runs in time $O(\log^6 n)$ which, although polylogarithmic, is larger than a $O(\log^4 n)$ -time algorithm for a maximal matching from [HKP01]. This increase in time complexity is due to the fact that we didn't succeed in parallelizing the computations to such a degree as it is done in [HKP01]. It should be emphasized though, that our algorithm and the maximal matching algorithm of Hańcówkiak, Karoński, and Panconesi share many common features. In particular both of them construct graph spanners in

blocks and use these spanners to compute matchings. One difference that increases the running time of our algorithm is such that unlike in [HKP01] where computations in blocks are done in parallel, we have to deal with $O(\log^2 n)$ blocks one by one in a sequential fashion.

The rest of the paper is structured as follows. In the next section, we present definitions and notation which are necessary to develop the algorithm. The last section of the paper contains the algorithm and the proof of its correctness.

2 Definitions and Notation

Because in the course of our algorithm an auxiliary multigraph is created we have to consider multigraphs instead of simple graphs. One important property of the obtained multigraphs is that the number of edges is always a polynomial in the number of vertices. Let M be a matching in a (multi)graph G . We say that a vertex v is M -saturated if v is an endpoint of some edge from M . An edge $e = \{u, v\}$ is M -saturated if either u or v is M -saturated.

Path P is M -alternating if it contains alternately edges from M and from $E \setminus M$. Path P of length $2k + 1$, $k \geq 0$, *augments* M if P is M -alternating and both ends of P are not M -saturated. Special role in the paper will be played by paths of length three augmenting M . A path is called an $(M, 3)$ -path if it augments M and has length three. A set of paths is called *independent* if every two paths from the set are vertex-disjoint. Next we define the notion of a substantial matching and a substantial set of paths.

Definition 2 *A matching M is γ -substantial in a multigraph $G = (V, E)$ if the number of edges of G that have an M -saturated endpoint is at least $\gamma|E|$.*

We will also use a notion of substantial set of paths. Let $\mathcal{P}_3(\mathcal{M})$ denote the set of all $(M, 3)$ -paths in G .

Definition 3 *Let M be a matching in G . A set \mathcal{P} of $(M, 3)$ -paths is called γ -path-substantial in G if the number of paths from $\mathcal{P}_3(\mathcal{M})$ that have a common vertex with some path in \mathcal{P} is at least $\gamma|\mathcal{P}_3(\mathcal{M})|$.*

The approach used to find a set of augmenting paths is based on the following strategy. First we find a maximal matching M using the procedure from [HKP01]. Then in the input graph G there will be three types of edges:

- (a) Edges that are in M .
- (b) Edges that have exactly one endpoint which is M -saturated (including edges that form a triangle based on an edge from M).
- (c) Edges that are not in M but have both endpoints M -saturated.

Note that edges from (c) do not belong to any $(M, 3)$ -path of G and therefore we can delete them. In the second main step of our procedure the original graph is reduced to a special four-layered form. Then we invoke a procedure that finds a set of $(M, 3)$ -paths in this layered graph, and finally translate the paths to the original graph G .

We shall start fixing more technical terminology by introducing a notion of a block. A bipartite (multi)graph $H = (A, B, E)$ is called a D -block if for every vertex $a \in A$,

$$\frac{D}{2} < \deg_H(a) \leq D.$$

A key concept which will be used in our approach is the concept of a spanner.

Definition 4 *Let $H = (A, B, E)$ be a D -block. An (α, β) -spanner from A to B is a subgraph $S = (A', B, E')$ of H such that the following conditions are satisfied.*

- (1) $|A'| \geq \alpha|A|$.
- (2) For every vertex $a \in A'$, $\deg_S(a) = 1$.
- (3) For every vertex $b \in B$, $\deg_S(b) < \frac{\beta}{D}\deg_H(b) + 1$.

In other words, a spanner is a collection of stars such that degrees of centers of stars are bounded. Note that spanners played an important role in designing an algorithm for finding a maximal matching in [HKP01]. In particular the following fact is proved in [HKP01].

Lemma 5 *Let $H = (A, B, E)$ be a simple graph which is a D -block and let $n = |A| + |B|$. There is a distributed algorithm that finds in $O(\log^3 n)$ steps an $(\frac{1}{2}, 16)$ -spanner.*

Yet another ingredients that we borrow from [HKP01] is a procedure that finds a γ -substantial matching in a bipartite multigraph. To find such a matching we invoke the procedure from [HKP01] (Lemma 4.7). Although in [HKP01] the procedure is formulated for simple graphs, it works for multigraphs with only minor changes. Since the procedure is long and complicated (in fact after $O(\log n)$ iterations of it, a maximal matching is easily obtained), we describe the main idea emphasizing the small changes that must be made in the multigraph case. Let $G = (L, R, F)$ be a bipartite multigraph, with $|L| = |R| = n$ and $|F| \leq n^C$ for some constant C . Then the bipartite multigraph G is split into D -blocks. Recall that a D -block is a bipartite multigraph (L_i, R, F_i) such that for every vertex $v \in L_i$, $D/2 < \deg(v) \leq D$. In [HKP01], G is split into $O(\log n)$ D -blocks for values $D = n/2^i$, $i = 0, \dots, \log n$. Here we make the first change in the multigraph case. Since the degree of any vertex in the multigraph is a number from 0 to n^C , we split our bipartite multigraph into $O(\log n)$ D -blocks for $D = n^C/2^i$, $i = 0, \dots, C \log n$. Next main step of the procedure from [HKP01] is to find substantial matchings (in parallel) in all

blocks and then to combine them. To find a substantial matching in a D -block a $(\frac{1}{2}, 16)$ -spanner is computed in the block and then the matching is obtained from this spanner. Finding a spanner is however not a trivial task and the main component of the procedure from Lemma 5 is a procedure SPLITTER which splits a simple graph into two graphs such that the degree of almost every vertex in the first graph is approximately one half the degree of the vertex in the original graph. The procedure splitter is iterated $O(\log n)$ times and a $(\frac{1}{2}, 16)$ -spanner is obtained. This is where we must make the second small change when we deal with multigraphs. To obtain a spanner in the multigraph, we must increase the number of iterations by a constant factor so that $1/2$ fraction of vertices from L_i are covered. In addition, the β constant in the third condition of the definition of (α, β) -spanner becomes a function of C . However as long as C is independent of n this will not change the asymptotic running time of the procedure that finds a substantial matching. Therefore we have the following lemma.

Lemma 6 *Let $G = (L, R, F)$ be a bipartite multigraph with $|L| = |R| = n$ such that $|F| \leq n^C$ for some constant C . Then for any $0 < \gamma < 1$ there is a distributed algorithm that finds in $O(\log^3 n)$ steps a γ -substantial matching in G .*

Recall that the main idea of our algorithm is first to find a maximal matching using the algorithm from [HKP01]. Then we obtain a virtual auxiliary graph which has a special layered structure. It turns out that the layered structure helps us to find a substantial set of $(M, 3)$ -paths in the layered graph. Finally, once the paths are found in the layered graph, we translate them back to the original graph. The layered graph will be a simple graph which has four layers of vertices. Graph G is called a $4L$ -graph if the vertex set of G is partitioned into four nonempty sets X_1, X_2, X_3, X_4 so that

- every vertex from X_1 is connected only with vertices from X_2 and every vertex from X_4 is connected only with vertices from X_3 ;
- $|X_2| = |X_3| = m$ and the edges between layers X_2 and X_3 form a perfect matching of size m .

Definition 7 *A $4L$ -graph $G = (X_1, X_2, X_3, X_4, E)$ is called a (D_1, D_2) -block if for every vertex $x \in X_2$,*

$$\frac{D_1}{2} < \deg_G(x) - 1 \leq D_1,$$

and for every vertex $x \in X_3$,

$$\frac{D_2}{2} < \deg_G(x) - 1 \leq D_2.$$

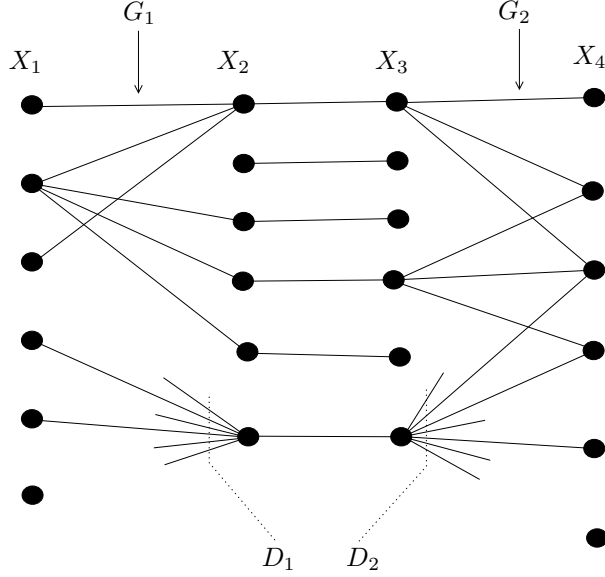


Fig. 1. A single (D_1, D_2) -block.

In the coming section we will also make use of graphs G_1 and G_2 defined below. Let $G_1 = G[X_1, X_2]$ be a graph induced by the first two layers, let $G_2 = G[X_3, X_4]$ be a graph induced by the last two layers, and let $m = |X_2| = |X_3|$ (see Figure 1).

3 Algorithm

In this section, we present the main algorithm that approximates the maximum matching. As noted in the introduction, the algorithm uses Theorem 1. Again let us review the main steps of our strategy. First, we find a maximal matching M in a graph G using the algorithm from [HKP01]. Note that, it seems to be difficult to find a substantial set of $(M, 3)$ -paths in graph G . This is the reason why after finding matching M , we reduce G to a virtual auxiliary graph G' which is a 4L-graph. Second, we consider the "largest" (D_1, D_2) -block in G' and find a substantial set of $(M, 3)$ -paths in the block. This set of paths is then translated back to the set P of $(M, 3)$ -paths in the original graph G . It turns out that after this translation phase the substantial property is preserved in "the image" of the block in graph G . We keep the set P , delete all paths that share a vertex with some path from P , and continue to compute independent paths in the largest block until the block is empty. Once the block is empty (this will happen after $O(\log n)$ iterations) we move to the "largest" nonempty block and repeat the above steps. Since $D_1 = n/2^i, D_2 = N/2^j$ where $i = 0, \dots, \log n, j = 0, \dots, \log n$, there are $O(\log^2 n)$ possible (D_1, D_2) -blocks. Consequently in the polylogarithmic number of steps we obtain a maximal set of independent $(M, 3)$ -paths. We divided

the algorithm into five procedures. In the first one (PROCEDURE PATHSINBLOCK) we compute a substantial set of paths in a (D_1, D_2) -block. Procedures REDUCE and TRANSLATE are used to construct the layered graph from G and to obtain a set of paths in graph G from the one found in a (D_1, D_2) -block. All three procedures are put together in procedure PATHS which contains a single iteration of the algorithm. Finally, INDEPENDENT-PATHS iterates over all possible blocks.

Next procedure finds a "substantial" set of independent paths in a (D_1, D_2) -block.

PROCEDURE PATHSINBLOCK

- (1) Using the algorithm from [HKP01] find a $(\frac{1}{2}, 16)$ -spanner S from X_3 to X_4 in $G_2 = G[X_3, X_4]$.
- (2) Construct the following auxiliary multi-graph $G'_1 = (X_1, X'_2)$.
 - For every star in the spanner S let $x_3(1), \dots, x_3(l)$ be the vertices in X_3 that have degree one in S and let $x_2(1), \dots, x_2(l)$ be the vertices in X_2 that are matched with $x_3(1), \dots, x_3(l)$ by the matching M between X_2 and X_3 . Create a *super-vertex* $s = s(x_2(1), \dots, x_2(l)) = \{x_2(1), \dots, x_2(l)\}$. The vertex set X'_2 contains all super-vertices.
 - For every vertex $x_1 \in X_1$ and every $x_2(i) \in s(x_2(1), \dots, x_2(l))$, put an edge between x_1 and the super-vertex $s = s(x_2(1), \dots, x_2(l))$ in G'_1 if x_1 and $x_2(i)$ are connected in G_1 .
- (3) Find a γ -substantial matching M' in G'_1 .
- (4) Extend (in a unique way) every edge of M' that contains a super-vertex to a path P of length three in the block using an edge of matching M in (X_2, X_3) and an edge of a star in a spanner S .

Let \mathcal{P} be the set of all paths P found by PROCEDURE PATHSINBLOCK. Note that paths in \mathcal{P} are independent and as we show next they form a substantial set of paths.

Lemma 8 *Let M_1 be the set of edges of G_1 that are contained in the set of paths \mathcal{P} and let M_2 be the set of edges of G_2 that are contained in \mathcal{P} . For any fixed $0 < \kappa < \gamma/4$ either M_1 is $(\gamma - 4\kappa)/4$ -substantial in G_1 or M_2 is $\kappa/16$ -substantial in G_2 .*

Proof.

Recall that $m = |X_2| = |X_3|$. Let $k = |M'|$ where M' is found in the third step of PROCEDURE PATHSINBLOCK and let $s^1 = (x_2^1(1), \dots, x_2^1(l_1)), \dots, s^k = (x_2^k(1), \dots, x_2^k(l_k))$ be the super-vertices that are saturated by M' . Note that the edges of the multi-graph G'_1 are in one-to-one correspondence with edges of G_1 . Let x^1, \dots, x^k denote vertices of X_2 that are contained in super-vertices s^1, \dots, s^k , ($x^i \in s^i$) which are such that x^i is the endpoint of an edge in G_1

that corresponds to an edge from M' that saturates s^i . In other words these are the vertices of X_2 that are saturated by M_1 . Finally let

$$W = \{x_2^i(j) : 1 \leq i \leq k, 1 \leq j \leq l_i\} \setminus \{x^1, \dots, x^k\}. \quad (1)$$

Thus W contains these vertices of X_2 that are contained in super-vertices and are not the endpoints of M_1 . The situation is illustrated in Figure 2.

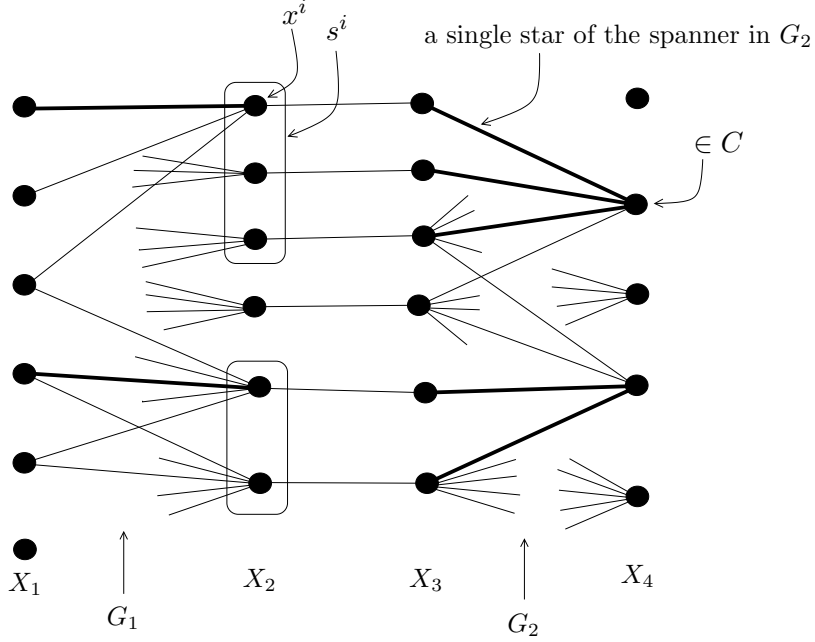


Fig. 2. The main argument.

Now we distinguish two cases:

Case 1: $|W| < \kappa m$.

In this case we show that M_1 is $(\gamma - 4\kappa)/4$ -substantial. Indeed, first note that the spanner S in G_2 found in the first step of the algorithm covers at least $\frac{m}{2}$ vertices of X_3 . Thus at least $\frac{m}{2}$ vertices of X_2 are contained in super-vertices. Since we operate in a (D_1, D_2) -block, we have

$$|E(G'_1)| \geq \frac{m}{2} \cdot \frac{D_1}{2} \geq \frac{|E(G_1)|}{4}.$$

Let us count the edges that are M_1 -saturated. Since M' is γ -substantial, at least $\gamma|E(G'_1)|$ edges are M' -saturated in G'_1 , not all of them though will correspond to M_1 -saturated edges in G_1 . We count these M' -saturated edges of G'_1 that do not correspond to M_1 -saturated edges in G_1 . Since $|W| < \kappa m$, there are at most

$$|W|D_1 < \kappa m D_1 \leq 4\kappa |E(G'_1)|$$

edges in G_1 that have an endpoint in W and possibly are not saturated by the final matching M_1 in G_1 . Therefore the number of edges that are saturated by M_1 is at least

$$(\gamma - 4\kappa)|E(G'_1)| \geq \frac{(\gamma - 4\kappa)}{4}|E(G_1)|$$

and M_1 is $(\gamma - 4\kappa)/4$ -substantial.

Case 2: $|W| \geq \kappa m$.

This case leads to a $\kappa/16$ -substantial matching in G_2 . Let C denote the set of vertices of X_4 that are the centers of the stars from S (see Figure 2). Then

$$\kappa m \leq |W| \leq \sum_{x \in C} (\deg_S(x) - 1) \leq 16 \sum_{x \in C} \frac{\deg_{G_2}(x)}{D_2}$$

where the last inequality follows from the fact that S is an $(\frac{1}{2}, 16)$ -spanner in G_2 . Therefore the number of edges that are M_2 -saturated is at least

$$\sum_{x \in C} \deg_{G_2}(x) \geq \frac{\kappa}{16} m D_2 \geq \frac{\kappa}{16} |E(G_2)|$$

and M_2 is $\kappa/16$ -substantial. \square

Lemma 9 *If M_1 is γ -substantial in G_1 or M_2 is γ -substantial in G_2 then the set of paths obtained by PROCEDURE PATHSINBLOCK is $\gamma/4$ -path-substantial in (D_1, D_2) -block.*

Proof.

Suppose that M_1 is γ -substantial in G_1 . Then the number of paths of length three in (D_1, D_2) -block that have a common vertex with paths obtained by the procedure is at least

$$\gamma |E(G_1)| \frac{D_2}{2} \geq \frac{\gamma}{4} m D_1 D_2$$

and clearly there are at most $m D_1 D_2$ paths of length three in a block. In the case when M_2 is γ -substantial, the proof is the same. \square Note that Lemma 9 implies that if we iterate procedure PROCEDURE PATHSINBLOCK in a (D_1, D_2) -block $O(\log n)$ times, each time deleting edges that are incident to selected paths and recomputing the block, then we will end up with a block without any paths of length three. Indeed, there are less than n^4 paths of length three in the whole graph and therefore in the block as well. In each iteration, we find a set of independent paths which is $\gamma/2$ -path-substantial, and so a constant fraction of paths will be deleted. If we iterate the process $O(\log n)$ times, there will be no paths in the block. Note however, that a block can become empty

due to yet another reason. Since we recompute the structure of the block after each iteration, it is possible (and likely) that degrees of some vertices will be significantly smaller after iterations and as a result these vertices will no longer be a part of the block. Since eventually we iterate over all possible blocks (starting from the largest) "dumped" vertices will be considered again once we move to smaller blocks.

Now we can explain how to handle a general graph G . In our main procedure we first reduce the situation in a general graph to one in a $4L$ -graph (reduction phase), then we invoke PROCEDURE PATHSINBLOCK in the largest nonempty block, and finally we translate the set of paths obtained by the procedure to the set of paths in the original graph G . Now we shall describe two procedures: REDUCE that obtains a $4L$ -graph from a general graph G and TRANSLATE that obtains independent augmenting paths in G from paths in the $4L$ -graph obtained by PATHSINBLOCK. The reduction phase is illustrated in Figure 3 and the translation in Figure 4. Note that although in Figure 3 (1) the graph already seems to be in the layered form, vertices of the graph in Figure 3 (1) do not know to which layer they belong and only in Figure 3 (4) vertices x^- know to be in the first layer, x^+ know to be in the fourth layer. Also, it could happen that for $\{m_1, m_2\}, \{n_1, n_2\} \in M$ with $m_1 < m_2, n_1 < n_2$ vertex x is connected with m_1 and is connected with n_2 .

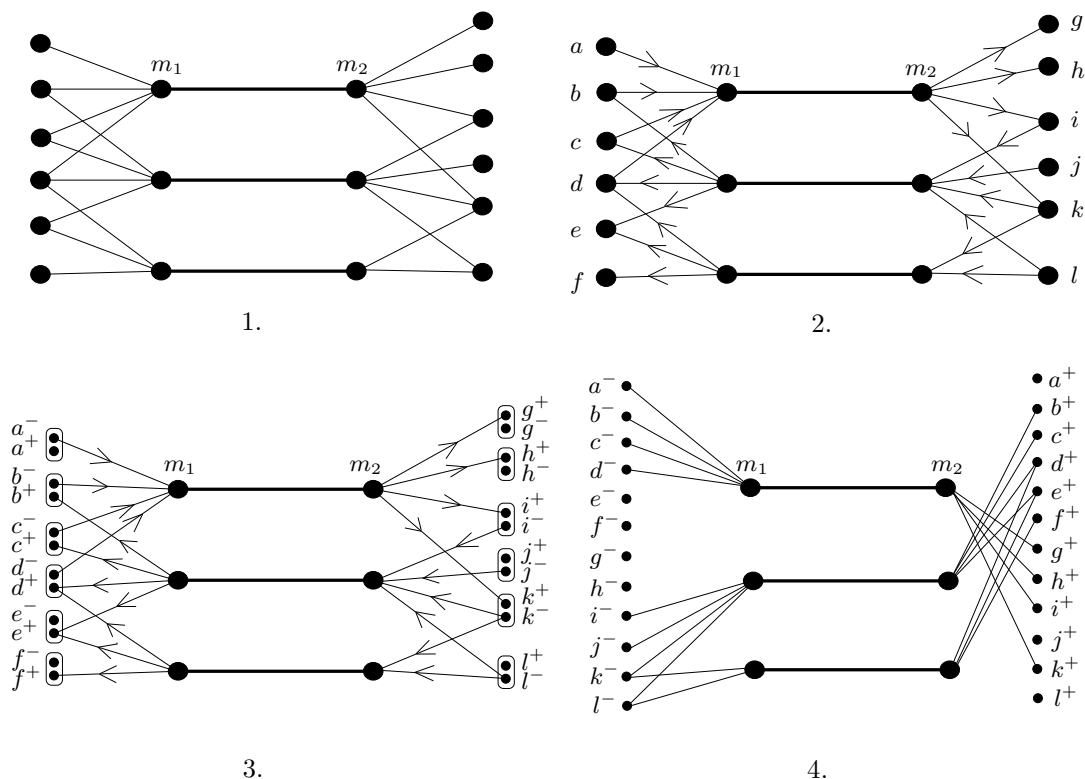


Fig. 3. The reduction.

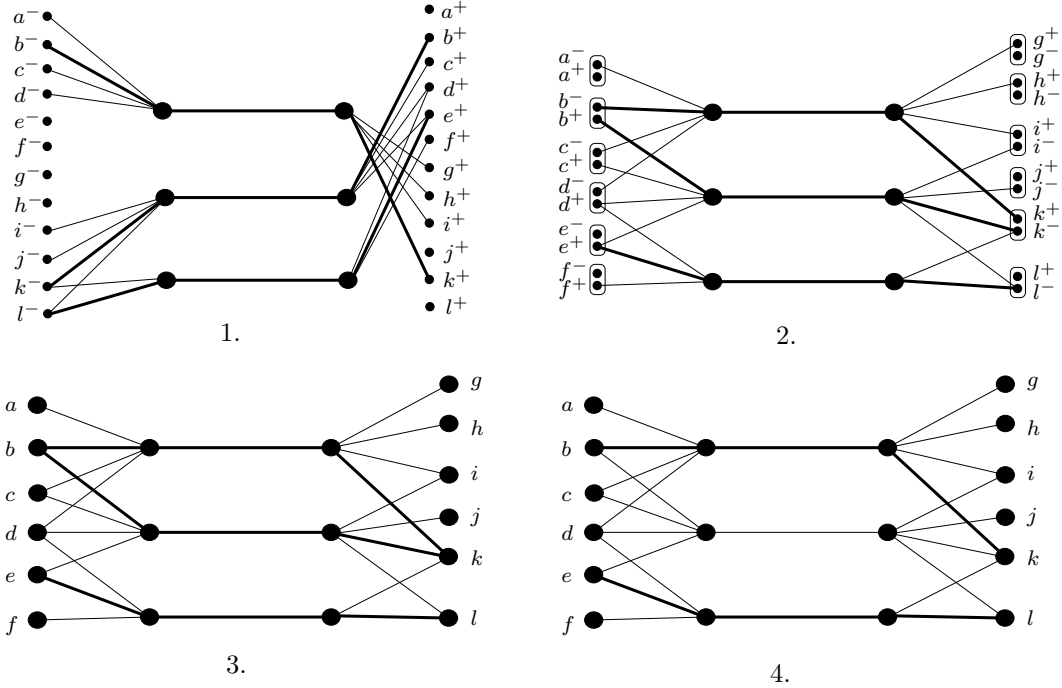


Fig. 4. The translation.

Next procedure reduces a graph G and a maximal matching M to 4L-graph G' . Recall that the main reason for this reduction is that we do not know how to find a substantial set of $(M, 3)$ -paths in G but we do know how to find a substantial set of $(M, 3)$ -paths in a block of a 4L-graph. Thus the goal of REDUCE is to create four sets of vertices V^-, V_1, V_2, V^+ such that M is the matching between V_1 and V_2 , vertices from V^- are connected only with vertices from V_1 , and vertices from V^+ are connected only with vertices from V_2 . Of course, in addition, the layered graph G' must have some properties that make it "similar" to G . In particular, one property that we must have is that the number of $(M, 3)$ -paths in G' will be a constant fraction of the number of $(M, 3)$ -paths in G . Also, $(M, 3)$ -paths in G' must correspond to $(M, 3)$ -paths in G and not to cycles. To obtain V^-, V_1, V_2, V^+ REDUCE does the following. First any edge of G which is not in a $(M, 3)$ -path is deleted. Then the triangles with one edge in M are destroyed by deleting one edge from each of them. This must be done carefully so that the number of $(M, 3)$ -paths in the layered graph does not decrease too much. To obtain V_1 and V_2 from M , REDUCE looks at the identifiers of the endpoint of edges in M and add a vertex to V_1 if its identifier is smaller than the identifier of the second end of an edge from M . Finally, V^- and V^+ are obtained by considering orientations of edges from $E \setminus M$ and splitting each unsaturated vertex v into two siblings v^- and v^+ .

PROCEDURE REDUCE

Given: Graph G and a maximal matching M in G .

- (1) For $e \in E(G)$ (in parallel) check if e is contained in at least one $(M, 3)$ -path. If it is not then delete e . In particular all edges from $E(G) \setminus M$ which have both endpoints M -saturated are deleted.
- (2) For every edge $m = \{m_1, m_2\} \in M$, with $ID(m_1) < ID(m_2)$, let T_m be the set of vertices in V such that every vertex from T_m is connected with m_1 and m_2 . Partition T_m into two groups $T_{m,1}$ and $T_{m,2}$ so that $||T_{m,1}| - |T_{m,2}|| \leq 1$. For every vertex $t \in T_{m,1}$ delete the edge $\{t, m_2\}$ from the graph, for every vertex $t \in T_{m,2}$ delete $\{t, m_1\}$. As a result, "new" graph G' does not have triangles based on edges from M (see Figure 5). From now we will operate on G' .
- (3) For every edge $m = \{m_1, m_2\} \in M$, with $ID(m_1) < ID(m_2)$, if $e \in E(G')$ and $e = \{v, m_1\}$ for some $v \neq m_2$ then give an orientation to e from v to m_1 , that is delete e and add arc (v, m_1) . If $e \in E(G')$ and $e = \{v, m_2\}$ for some $v \neq m_1$ then give an orientation to e from m_2 to v , that is delete e and add arc (m_2, v) .
- (4) For every vertex $v \in G \setminus V(M)$ split v into two siblings v^- and v^+ , where v^- inherits all the arcs that start in v , v^+ inherits arcs that end in v . Finally, ignore the orientation on the edges. As a result we obtain a 4L-graph $H = (V^-, V_1(M), V_2(M), V^+)$, where $V^- = \{v^-, v \in G \setminus V(M)\}$, $V^+ = \{v^+, v \in G \setminus V(M)\}$, and $V_1(M), V_2(M)$ are corresponding endpoints of M that is if $m = \{m_1, m_2\} \in M$ and $ID(m_1) < ID(m_2)$ then $m_1 \in V_1(M), m_2 \in V_2(M)$.

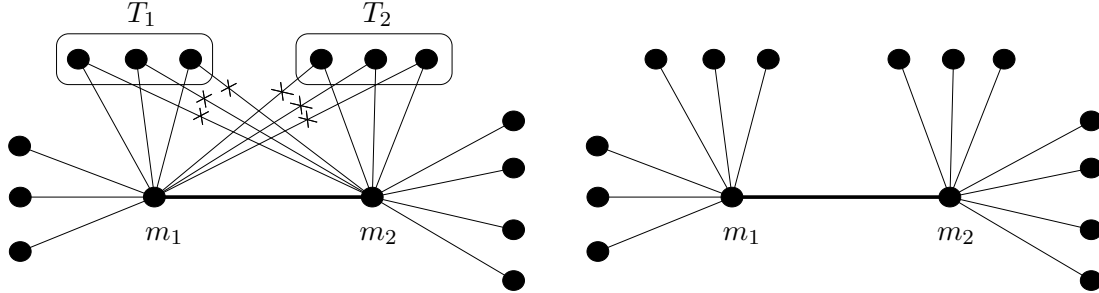


Fig. 5. Removing the triangles.

Let $G(D_1, D_2)$ denote the subgraph of G that consists of edges from M that appear in a (D_1, D_2) -block of H and edges of G that are saturated by them. Let $G'(D_1, D_2)$ be a subgraph of $G(D_1, D_2)$ with deleted edges in (2) of REDUCE.

PROCEDURE TRANSLATE

Given: Set \mathcal{P}_H of independent $(M, 3)$ -paths in a (D_1, D_2) -block of H .

- (1) Identify vertices v^- and v^+ that correspond to one vertex $v \in V(G)$. As a result we obtain from \mathcal{P}_H cycles and paths in graph G that consist of paths

of length three that augment M . Note however that we do not obtain triangles in G as all triangles based on edges from M were destroyed in (2) of REDUCE.

- (2) Treat $(M, 3)$ -paths as edges between endpoints of these paths. Now the problem of selecting a substantial set of independent $(M, 3)$ -paths in G is reduced to the one of finding a substantial set of independent edges. Invoke an algorithm from [HKP01] to obtain set \mathcal{P}_G of independent (in G) $(M, 3)$ -paths with the property that at least $1/4$ of $(M, 3)$ -paths in $G'(D_1, D_2)$ that have a common vertex with \mathcal{P}_H have also common vertex with some path from \mathcal{P}_G .

Our next procedure is essentially one iteration of the main algorithm.

PROCEDURE PATHS

Given: Two numbers, D_1, D_2 , and graph G .

- (1) Use REDUCE to obtain 4L-graph H from G .
- (2) Consider a (D_1, D_2) -block and invoke PATHSINBLOCK in the (D_1, D_2) -block to find a set \mathcal{P}_H of independent paths in H .
- (3) Use TRANSLATE to obtain set \mathcal{P}_G of independent paths in G .

By Lemma 8 and Lemma 9 set \mathcal{P}_H is γ -path-substantial, for some $0 < \gamma < 1$ in the (D_1, D_2) -block. We claim that \mathcal{P}_G is $\gamma/16$ - path-substantial in $G(D_1, D_2)$.

Lemma 10 *If \mathcal{P}_H is γ -path-substantial in a (D_1, D_2) -block then \mathcal{P}_G is $\gamma/16$ -path-substantial in $G(D_1, D_2)$.*

Proof.

Recall that G' denotes a graph obtained from G by deleting triangles based on M in the second step of REDUCE . Note that $(M, 3)$ -paths in the (D_1, D_2) -block of H are in one-to-one correspondence with $(M, 3)$ -paths in graph $G'(D_1, D_2)$. Thus, to prove the lemma, it is enough to argue the following two facts:

- (1) \mathcal{P}_G is $\gamma/4$ - path-substantial in $G'(D_1, D_2)$,
- (2) the total number of $(M, 3)$ -paths in $G'(D_1, D_2)$ is at least one fourth times the total number of $(M, 3)$ -paths in $G(D_1, D_2)$.

To prove the first part, note that in the second step of PROCEDURE TRANSLATE we obtain the set \mathcal{P}_G such that at least $\gamma/4$ fraction of $(M, 3)$ -paths in $G'(D_1, D_2)$ have at least one vertex in common with some path from \mathcal{P}_G . Therefore, \mathcal{P}_G is $\gamma/4$ -path-substantial in $G'(D_1, D_2)$.

To show part (2), we argue that for every edge $m \in M$ the number of $(M, 3)$ -paths in $G'(D_1, D_2)$ that contain m is at most four times the number of $(M, 3)$ -

paths in $G(D_1, D_2)$ that contain m . Consider an edge $m = \{m_1, m_2\}$ of M and suppose that there are $2a$ triangles based on m in $G(D_1, D_2)$ (case $2a + 1$ leads to the same computations). Let $\deg_G(m_1) = 2a + b$, $\deg_G(m_2) = 2a + c$. Then $\deg_{G'}(m_1) = a + b$, $\deg_{G'}(m_2) = a + c$ and there are $(a + b)(a + c)$, $(M, 3)$ -paths containing m in G' . On the other hand, there are at most $(2a + b)(2a + c)$, $(M, 3)$ -paths containing m in $G(D_1, D_2)$. Clearly $(a + b)(a + c) \geq (2a + b)(2a + c)/4$, and so the number of $(M, 3)$ -paths in $G'(D_1, D_2)$ is at least one fourth of the number of $(M, 3)$ -paths in $G(D_1, D_2)$.

Finally, observe that (1) and (2) prove the lemma. Let $\mathcal{P}(G, 3)$ ($\mathcal{P}(G', 3)$) denote the set of $(M, 3)$ -paths in $G(D_1, D_2)$ (in $G'(D_1, D_2)$). By (2),

$$|\mathcal{P}(G', 3)| \geq \frac{|\mathcal{P}(G, 3)|}{4},$$

thus a set which is γ -path-substantial in $G'(D_1, D_2)$ is $\gamma/4$ -path-substantial in $G(D_1, D_2)$. From (1) we see that \mathcal{P}_G is $\gamma/4$ -path-substantial in $G'(D_1, D_2)$. \square Before continuing with our main algorithm, let us say precisely how we modify the graph G after a set of independent paths \mathcal{P} is found.

- (1) We delete all edges that share at least one vertex with some path from \mathcal{P} .
- (2) We delete all edges that do not belong to any $(M, 3)$ -path in G (such edges may appear after step (1)).

Main algorithm iterates procedure PATHS, after each iteration it modifies graph G and matching M . Let $\xi = 1/2$ be the constant used in PATHSINBLOCK to find a ξ -substantial matching in the bipartite multigraph of the block, let $\kappa = \xi/16$ be the constant from Lemma 8, and let γ be such that the set of paths is γ -substantial in $G(D_1, D_2)$ after PATHS is invoked (see Lemma 10 and Lemma 9). Note that γ does not depend on the values of D_1 and D_2 and in fact $\gamma = \frac{1}{4(16)^3}$.

PROCEDURE INDEPENDENT-PATHS

Given: Graph G ; constants $\xi := 1/2$, $\kappa := \xi/16$ to be used in PATHSINBLOCK, γ (Lemma 10), and $c := 1/\gamma$.

- (1) Compute a maximal matching M using a procedure from [HKP01].
- (2) Delete all edges in G that do not appear in any $(M, 3)$ -path. In particular edges from $E(G) \setminus M$ that have both endpoints M -saturated are deleted.
- (3) for $i = 0$ to $\lceil \log n \rceil$ do:
 - (a) $D_1 = \frac{n}{2^i}$, $D_2 = \frac{n}{2^i}$
 - (b) iterate $4 \log_c n$ times:
 - invoke PROCEDURE PATHS with D_1, D_2 in graph G and modify G and M

- (c) for $j = i + 1$ to $\lceil \log n \rceil$ do:
 - (i) let $D_1 = \frac{n}{2^j}$ and $D_2 = \frac{n}{2^i}$
 - (ii) iterate $4 \log_c n$ times:
 - invoke PROCEDURE PATHS with D_1, D_2 in graph G and modify G and M
 - (iii) let $D_1 = \frac{n}{2^i}$ and $D_2 = \frac{n}{2^j}$
 - (iv) iterate $4 \log_c n$ times:
 - invoke PROCEDURE PATHS with D_1, D_2 in graph G and modify G and M

Note that in PROCEDURE INDEPENDENT-PATHS we iterate over all $O(\log^2 n)$ possible blocks. In each block we iterate PROCEDURE PATHS $O(\log n)$ times so that the block is empty (has no $(M, 3)$ -paths) after these iterations. It is important to notice that we move from one block to another in a specific way. We start with the block that has the highest possible sum of parameters D_1, D_2 and then we consider smaller sums. Note that there are at most two blocks that have given sum of D_1 and D_2 . These two blocks are emptied one after another in iterations of steps (ii) followed by (iv). It is easy to notice that once a block is emptied it is not possible that in future iterations (after modifications) we obtain an edge of M that belongs to this block.

Theorem 11 PROCEDURE INDEPENDENT-PATHS finds a maximal set of independent $(M, 3)$ -paths in graph G in $O(\log^6 n)$ steps.

Proof.

Let us first argue that the running time is $O(\log^6 n)$. Indeed, we have $O(\log^2 n)$ iterations over all possible blocks. We invoke PROCEDURE PATHS in one block $O(\log n)$ times, the running time of PROCEDURE PATHS is $O(\log^3 n)$ and comes from finding a spanner (Lemma 5) and a substantial matching (Lemma 6).

Next, let us notice that once a (D_1, D_2) -block is emptied in an iteration then in future iterations (after modifications) there will be no edge of M that is classified to belong to the (D_1, D_2) -block. We iterate over blocks with a highest sum of D_1, D_2 down to the lowest sum and we never add any edges to the graph. Consequently, if an edge belongs to a block (B_1, B_2) with $B_1 + B_2 < D_1 + D_2$ at the time of working on (D_1, D_2) -block then it can never be considered as an edge of a (D_1, D_2) -block. In addition, because of the definition of the degree constrains in a block, if an edge is in a (D_2, D_1) -block then it cannot be considered as an edge of the (D_1, D_2) -block .

Next, we shall show that after $O(\log n)$ iterations in a (D_1, D_2) -block, the block is empty. By Lemma 10, \mathcal{P}_G is γ -path-substantial (for some $0 < \gamma < 1$, in fact $\gamma = \frac{1}{4(16)^3}$) in $G(D_1, D_2)$. There are at most n^4 paths in G and so there are at most n^4 , $(M, 3)$ -paths in graph $G(D_1, D_2)$. In each iteration we

destroy at least γ fraction of them when modifying graph G and matching M . Therefore, after $4 \log_c n$ (with $c = 1/\gamma$) iterations there will be no $(M, 3)$ -path in $G(D_1, D_2)$.

Finally, observe that the set \mathcal{P} is an independent set of paths. Indeed, if \mathcal{P}_G is a set of independent $(M, 3)$ -paths found in one iteration then in the modification step we destroy all $(M, 3)$ -paths that have a common vertex with paths from \mathcal{P}_G . Therefore, $(M, 3)$ -paths found in next iterations will be independent of paths from \mathcal{P}_G . \square

References

- [FGHP93] T. Fischer, A. V. Goldberg, D. J. Haglin, S. Plotkin, *Approximating matchings in parallel*, Information Processing Letters, 1993, 46, pp. 115-118.
- [Li92] N. Linial, *Locality in distributed graph algorithms*, SIAM Journal on Computing, 1992, 21(1), pp. 193-201.
- [HKP01] M. Hańćkowiak, M. Karoński, A. Panconesi, *On the distributed complexity of computing maximal matchings*, SIAM J. Discrete Math. Vol.15, No.1, pp. 41-57.
- [AGLP89] B. Awerbuch, A.V. Goldberg, M. Luby, and S. Plotkin, *Network decomposition and locality in distributed computing*, Proceedings of the 30th Symposium on Foundations of Computer Science (FOCS 1989), pp. 364-369.