

# Distributed algorithm for better approximation of the maximum matching\*

A. Czygrinow<sup>†</sup>, M. Hańckowiak<sup>‡</sup>

November 3, 2005

## Abstract

Let  $G$  be a graph on  $n$  vertices that does not have odd cycles of lengths  $3, \dots, 2k - 1$  with  $k \geq 2$ . We give a deterministic distributed algorithm which finds in a poly-logarithmic (in  $n$ ) number of rounds a matching  $M$ , such that  $|M| \geq (1 - \alpha)m(G)$ , where  $m(G)$  is the size of a maximum matching in  $G$  and  $\alpha = \frac{1}{k+1}$ .

## 1 Introduction

A matching  $M$  in graph  $G = (V, E)$  is a subset of  $E$  such that no two edges from  $M$  share an endpoint. A matching  $M$  is called *maximal* if there is no matching  $M'$  such that  $M$  is a proper subset of  $M'$ . A matching  $M$  is called *maximum* if its size,  $|M|$ , is the largest possible. Let  $m(G)$  be the size of a maximum matching in  $G$ , that is  $m(G) = \max\{|M| : M \text{ is a matching in } G\}$ .

In this paper we are interested in designing an efficient distributed algorithm which finds in a graph  $G$  a matching  $M$  of size which is approximately  $m(G)$ . We consider the distributed model of computations introduced by Linial in [Li92]. In this model a distributed network is represented by an undirected graph. Vertices of the graph correspond to processors and edges to communication links between them. The network is synchronous and computations proceed in rounds. In a single round each vertex sends messages to its neighbors, receives messages from its neighbors, and performs some local computations. Neither the amount of local computations nor the lengths of messages are restricted in any way. Similarly as in the case of parallel algorithms, we will desire that the number of rounds is poly-logarithmic in  $|V(G)|$  and will call an algorithm *efficient* if it solves a given problem in the number of rounds which is poly-logarithmic in  $|V(G)|$ .

Designing efficient distributed algorithms faces many challenges. Maybe the most fundamental one lies in the fact that the topology of  $G$  impacts the extent to which of processors can communicate. In particular, in  $k$  steps, a vertex  $v$  can communicate only with vertices within distance  $k$  of  $v$  and so in a small number of rounds only vertices which are with close proximity to  $v$  can exchange information with  $v$ . Consequently processors are confined to their local subgraphs and based on that local information a global function of  $G$  must be found. The distributed model of computations is completely different than the massively parallel PRAM model. In the former model the processors power is irrelevant and the communication between nodes determines the complexity of an algorithm. In the latter, the processors can freely exchange information in the course of computations by means of shared memory. Unlike the case of PRAM model, only very few classical problems are known to admit efficient distributed

---

\*Research supported by KBN grant no. 7 T11C 032 20. Preliminary version of this paper appeared in proceedings of COCOON 2003.

<sup>†</sup>Department of Mathematics and Statistics, Arizona State University, Tempe, AZ,85287-1804, USA, andrzej@math.la.asu.edu.

<sup>‡</sup>Faculty of Mathematics and Computer Science, Adam Mickiewicz University, Poznań, Poland, mhanckow@main.amu.edu.pl.

algorithms. In particular even the maximal independent set (MIS) problem eludes efficient distributed solution in general graphs. In contrast, the PRAM algorithms for the MIS problem are well-known and in particular, a celebrated algorithm of Luby [Lu86] gives an efficient deterministic parallel algorithm. Luby's algorithm also gives an efficient parallel algorithm for the maximal matching problem and, not surprisingly, there are even faster parallel algorithms for the maximal matching problem (see [Ha95], [Ch95]). Algorithms from [Ha95] and [Ch95] as well as Luby's algorithm are specific to the PRAM model and their implementation in the distributed model is not possible. An approximation of the maximum matching can be easily accomplished by invoking the Luby's algorithm in the auxiliary graph of  $G$  with vertices that correspond to  $M$ -augmenting paths and edges between vertices whenever the corresponding paths intersect in  $G$ .

An efficient distributed algorithm for the maximal matching problem was given only recently by Hańkowiak, Karoński, and Panconesi in [HKP01]. It is their work which motivates the line of research described below. The algorithm from [HKP01] finds a maximal matching  $M$  which by virtue of maximality has  $|M| \geq m(G)/2$ . Using the techniques from [HKP01], Czygrinow, Hańkowiak, and Szymańska [CHS02] designed an efficient distributed algorithm that finds a matching  $M$  with  $|M| \geq \frac{2m(G)}{3}$ . In this paper, we give a distributed algorithm that finds an even better approximation of the maximum matching in the case when graph does not have short odd cycles. Let  $og(G)$  denote the length of the shortest odd cycle in  $G = (V, E)$  (if  $G$  is bipartite then we set  $og(G) := |V| + 1$ ).

**Theorem 1.1** *For every integer  $k > 1$  there exist a number  $D(k)$  and a distributed algorithm which in a graph  $G$  on  $n$  vertices with  $og(G) \geq 2k - 1$  finds a matching  $M$  such that*

$$|M| \geq \left(1 - \frac{1}{k+1}\right) m(G).$$

*The algorithm runs in  $O(\log^{D(k)} n)$  rounds.*

For a matching  $M$  in graph  $G$ , a path of length  $2l - 1$  is called  $M$ -augmenting if it has length  $2l - 1$ , edges alternate between  $M$  and  $E(G) \setminus M$ , and both endpoints are  $M$ -free (see the next section for necessary definitions). Main strategy of the algorithm from Theorem 1.1 is to find a matching  $M$  with no  $M$ -augmenting paths of short lengths. It is easy to see that such a matching must have the size which is close to  $m(G)$ :

**Theorem 1.2** [HK73] *Let  $M$  be a matching such that there are no  $M$ -augmenting paths of lengths  $1, 3, 5, \dots, 2k - 1$ . Then*

$$|M| \geq \left(1 - \frac{1}{k+1}\right) m(G).$$

Note that the assumption in Theorem 1.2 that there are no  $M$ -augmenting paths of length one means that  $M$  is maximal. A general idea of the algorithm from Theorem 1.1 is to find a maximal matching  $M$  and then to improve  $M$  by finding a collection of pair-wise disjoint  $M$ -augmenting paths and for each path  $P$  an substituting the edges from  $M \cap E(P)$  with edges from  $E(P) \setminus M$ . More precisely, the algorithm iterates with  $l = 2, \dots, k$  and in the  $l$ th iteration it finds a maximal set (with respect to inclusion) of pair-wise disjoint  $M$ -augmenting paths of length  $2l - 1$  each. Once a maximal set of paths is found, the edges in paths that belong to  $M$  and those that are not in  $M$  are exchanged. It is easy to see that after this exchange new matching  $M'$  does not have  $M'$ -augmenting paths of lengths which are less than or equal to  $2l - 1$ . Consequently, by Theorem 1.2,  $|M'|$  is at least  $\frac{l}{l+1} m(G)$ . Finding a maximal set of pair-wise disjoint  $M$ -augmenting paths is however not trivial and an algorithm that finds such a set is indeed the central result of this paper. A general idea of this algorithm is as follows. An auxiliary graph constructed from  $G$  and matching  $M$  is considered. Using the spanners technique from [HKP01] a maximal set of pair-wise disjoint  $M$ -augmenting paths is found in the auxiliary graph (procedure MAXIMAL-PATHS-IN-LAYERED-GRAPH). Structure of the auxiliary graph is critical here and the algorithms uses it extensively to find the paths. However  $M$ -augmenting paths in the auxiliary

graph must be translated back to  $M$ -augmenting paths in the original graph  $G$  and it is this place of the algorithm where the assumption about the odd girth,  $og(G)$ , is used. Indeed, since  $G$  does not have short odd cycles, we can prove that paths in the auxiliary graph correspond to paths (not to cycles) in graph  $G$ .

Finally let us note that distributed algorithms can be described in a few different ways. We shall follow conventions from [Pe02] and recent research papers in the area (for example [HKP01]) and give a high-level description to keep the algorithms as simple as possible. Each step of our algorithms can be implemented in a constant number of rounds in the distributed model of computations.

The rest of the paper is structured as follows. In the next section, we introduce necessary notation and auxiliary facts. Section 3 contains the procedure that finds a maximal set of pair-wise disjoint  $M$ -augmenting paths in the auxiliary graph. In Section 4, we describe the translation method and the main algorithm.

## 2 Definitions and Notation

In this section, we introduce necessary definitions and fix some notation. Let  $M$  be a matching in graph  $G = (V, E)$ . We say that a vertex  $v$  is  $M$ -saturated if  $v$  is an endpoint of some edge from  $M$ , otherwise  $v$  is called  $M$ -free. An edge  $e = \{u, v\}$  is  $M$ -saturated if either  $u$  or  $v$  or both are  $M$ -saturated.

We say that  $P$  is an  $M$ -alternating path if the edges of  $P$  alternate between these from  $M$  and these from  $E \setminus M$ . A path  $P$  of length  $2k - 1$ , where  $k \geq 1$ , *augments*  $M$  if  $P$  is  $M$ -alternating and both ends of  $P$  are  $M$ -free.

**Definition 2.1** *Let  $M$  be a matching and let  $k$  be a positive integer. A path is called an  $(M, 2k - 1)$ -path if it augments  $M$  and has length  $2k - 1$ .*

Paths are called *disjoint* if they are pair-wise vertex-disjoint. Next we define two important concepts: a  $\gamma$ -substantial matching and a  $\gamma$ -substantial set of paths.

**Definition 2.2** *Let  $0 < \gamma \leq 1$ . A matching  $M$  is  $\gamma$ -substantial in graph  $G = (V, E)$  if the number of  $M$ -saturated edges in  $G$  is at least  $\gamma|E|$ .*

Note that if  $\gamma = 1$  then all edges of  $G$  are  $M$ -saturated and so the matching  $M$  is maximal. In [HKP01] (see Lemma 3.1 on page 45 and Lemma 4.8 on page 50) an algorithm which finds a  $1/2$ -substantial matching is given. If this procedure is iterated  $O(\log \frac{1}{1-\gamma})$  times then a  $\gamma$ -substantial matching is obtained.

**Lemma 2.3** [HKP01] *Let  $n$  be the number of vertices of a graph. For any  $0 < \gamma < 1$  there is a distributed algorithm that finds in  $O(\log(\frac{1}{1-\gamma}) \log^3 n)$  rounds a  $\gamma$ -substantial matching.*

Note that Lemma 2.3 with  $\gamma = 1/2$  is the main component of the algorithm for the maximal matching from [HKP01]. Indeed once a  $\frac{1}{2}$ -substantial matching  $M$  is found then the number of  $M$ -saturated edges is a constant fraction of  $|E|$ . After deleting them, we can repeat the process and find a  $\frac{1}{2}$ -substantial matching in the new graph. Since at least half of the total number of edges is deleted in each step, after  $2 \log n$  steps there will be no edges in the graph. The notion of a  $\gamma$ -substantial matching extends in a natural way to a  $\gamma$ -substantial set of paths.

**Definition 2.4** *Let  $0 < \gamma \leq 1$ . For matching  $M$  and positive integer  $k$  let  $\mathcal{M}$  be the set of all  $(M, 2k - 1)$ -paths in graph  $G$ . A set  $\mathcal{P}$  of  $(M, 2k - 1)$ -paths is  $\gamma$ -substantial if the number of  $(M, 2k - 1)$ -paths that have at least one common vertex with some path from  $\mathcal{P}$  is at least  $\gamma|\mathcal{M}|$ .*

We will pose for a moment and outline a general strategy of our algorithm and how the above concepts are used in it. The algorithm iterates with  $i = 1, \dots, k$  and in the  $i$ th iteration it finds a maximal (with

respect to inclusion) set of disjoint  $(M, 2i - 1)$ -paths in graph  $G$  and augments them. In each iteration, the algorithm proceeds in three major steps. First, from graph  $G$  and matching  $M$ , an auxiliary graph is obtained. The graph has a special layered structure (Procedure REDUCE). Second, the algorithm finds a maximal set of paths in the auxiliary graph (Procedure MAXIMAL-PATHS-IN-LAYERED-GRAPH) and finally it translates paths from the auxiliary graph back to graph  $G$  (Procedure TRANSLATE). It is finding a maximal set of paths in the auxiliary graph that requires most of the work and which uses the concept of spanners which we will now introduce.

A bipartite graph  $H = (A, B, E)$  with the vertex set  $A \cup B$  and the edge set  $E$  is called a  $D$ -block if for every vertex  $a \in A$ ,

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

Spanners are special subgraphs of  $D$ -blocks.

**Definition 2.5** An  $(\alpha, \beta)$ -spanner of a  $D$ -block  $H = (A, B, E)$  (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  $b \in B$ ,

$$\deg_S(b) < \frac{\beta}{D} \deg_H(b) + 1. \quad (1)$$

In other words, a spanner is a family of stars such that degrees of the centers of stars satisfy (1). Spanners played an important role in the maximal matching algorithm from [HKP01] and are a central tool in our algorithm. In particular we will use the following fact from [HKP01] (see Theorem 5.5 on page 55).

**Lemma 2.6** [HKP01] Let  $H = (A, B, E)$  be a  $D$ -block and let  $n = |A| + |B|$ . There is a distributed algorithm that finds in  $O(\log^3 n)$  rounds a  $(\frac{1}{2}, 16)$ -spanner.

The main component of our algorithm is a procedure which finds a  $\gamma$ -substantial set of disjoint  $(M, 2k - 1)$ -paths. Finding such a set seems to be very difficult if a graph  $G$  is arbitrary. On the other hand if a graph has a layered structure then we can find a  $\gamma$ -substantial set of disjoint  $(M, 2k - 1)$ -paths using the spanners technique. This impacts our strategy. We will reduce graph  $G$  to an auxiliary, layered graph and exploit the structure of the auxiliary graph to find the set of paths and then translate the paths back to  $G$ . The auxiliary graph has a special layered structure with an even number of layers. In addition, the number of layers does not depend on  $n$  but on  $k$  only.

**Definition 2.7** Graph  $G$  is called a  $2L$ -layered graph if the vertex set of  $G$  can be partitioned into sets  $X_1, \dots, X_{2L}$  so that

- For every  $i = 1, \dots, 2L$ ,  $X_i$  is an independent set.
- For  $i, j = 1, \dots, 2L$ , there are no edges between sets  $X_i$  and  $X_j$  whenever  $|i - j| \geq 2$ .
- For  $i = 1, \dots, L - 1$ ,  $|X_{2i}| = |X_{2i+1}|$  and the edges between  $X_{2i}$  and  $X_{2i+1}$  form a perfect matching.

If  $G$  is a  $2L$ -layered graph with layers  $X_1, \dots, X_{2L}$  then we will write  $G = (X_1, X_2, \dots, X_{2L})$ . In addition, for  $i = 1, \dots, 2L - 1$ , we denote by  $G_i = G[X_i, X_{i+1}]$  the bipartite graph induced by  $X_i, X_{i+1}$ . Thus, for even  $i$ ,  $G_i$  is a perfect matching between  $X_i$  and  $X_{i+1}$ . We will also let  $G_{left}$  be the subgraph of  $G$  induced by  $(X_1, \dots, X_{2L-2})$  and, to make the notation uniform, we will set  $G_{right} = G_{2L-1}$ .

**Definition 2.8** Let  $G = (X_1, X_2, \dots, X_{2L})$  be a layered graph and let  $M = \bigcup_{i=1}^{L-1} E(G_{2i})$ . Then  $G$  is called a  $(D_1, D_2)$ -block if

- For every  $v \in X_{2L-1}$ ,  $d_{G_{right}}(v) \in [\frac{D_2}{2}, D_2]$ .
- For every  $v \in X_{2L-2}$  the number of  $(M, 2L-3)$ -paths with  $v$  as one endpoint and the other endpoint in  $X_1$  is larger than  $\frac{D_1}{2}$  and is at most  $D_1$ .

We will end this section by presenting the modification procedure which will be often used in the course of computations. We will repeatedly modify a graph with respect to a set of paths computed by an algorithm.

---

PROCEDURE MODIFY

*Input:* Graph  $G$  with matching  $M$ , positive integer  $k$ , set of  $(M, 2k-1)$ -paths  $\mathcal{P}$ .

*Output:* Graph  $G'$ .

1. For every edge  $e \in E(G)$ : If  $e$  is incident to a path from  $\mathcal{P}$  then remove  $e$  from  $G$ . In particular all edges in the paths from  $\mathcal{P}$  are removed.
  2. For every edge  $e$ : If there is no  $(M, 2k-1)$ -path which contains  $e$  then remove  $e$ . Edges of this sort can appear after the first step is executed. Let  $G'$  be the resulting graph.
- 

### 3 Maximal set of paths in a layered graph

In this section, we give an algorithm that finds a maximal set of disjoint  $M$ -augmenting paths in a layered graph. As mentioned before, layered graph is an auxiliary graph obtained from an original graph  $G$  and matching  $M$  structure of which allows to find paths in a much easier way than in graph  $G$ . The algorithm for finding a maximal set of disjoint  $M$ -augmenting paths consists of two main procedures: SUBSTANTIAL-PATHS-IN-BLOCK and MAXIMAL-PATHS-IN-LAYERED-GRAPH. The first procedure finds a set of disjoint  $M$ -augmenting paths in a  $(D_1, D_2)$ -block. The procedure is recursive and in the recursive step it invokes MAXIMAL-PATHS-IN-LAYERED-GRAPH in a layered graph with a smaller number of layers. The second procedure, MAXIMAL-PATHS-IN-LAYERED-GRAPH, iterates over all  $(D_1(i), D_2(j))$ -blocks with  $D_1(i) = \frac{n^{L-1}}{2^i}$ ,  $D_2(j) = \frac{n}{2^j}$  where  $i = 1, \dots, \lceil (L-1) \log n \rceil$ ,  $j = 1 \dots, \lceil \log n \rceil$ . In each block it invokes SUBSTANTIAL-PATHS-IN-BLOCK  $O(\log n)$  times modifying the block after each iteration. As a result, after these  $O(\log n)$  iterations the block will be the empty graph. Since every  $M$ -augmenting path in the layered graph is a path of some  $(D_1, D_2)$ -block the union of the sets of paths will be maximal in  $G$ .

Let  $G = (X_1, \dots, X_{2L})$  be a  $(D_1, D_2)$ -block. Recall that  $G_i$  denotes the bipartite graph  $G[X_i, X_{i+1}]$  and so in the case when  $i$  is even,  $G_i$  is a perfect matching between  $X_i$  and  $X_{i+1}$ . To find a substantial set of disjoint paths in  $G$ , SUBSTANTIAL-PATHS-IN-BLOCK finds a  $(\frac{1}{2}, 16)$ -spanner in  $G_{right} = G_{2L-1}$  with centers of stars in  $X_{2L}$ . For each star  $(z, x(1), \dots, x(k))$ , with  $z \in X_{2L}$  and  $x(1), \dots, x(k) \in X_{2L-1}$  consider vertices  $x'(1), \dots, x'(k) \in X_{2L-2}$  which are matched with  $x(1), \dots, x(k)$  by edges of the matching from  $G_{2L-2}$  (see Figure 1). In the next step of the procedure new layered graph  $G'_{left}$  is constructed (in fact  $G'_{left}$  is a multi-graph). Graph  $G'_{left}$  has  $2L-2$  layers  $(X_1, X_2, \dots, X_{2L-3}, X')$  with the new layer  $X'$  containing the so-called *super-vertices* which correspond to the stars of the spanner found before. Then procedure calls MAXIMAL-PATHS-IN-LAYERED-GRAPH with  $G'_{left}$  as an input graph. As a result a maximal set of disjoint  $(M, 2L-3)$ -paths is obtained. Paths from the set are then extended to  $(M, 2L-1)$ -paths using the matching in  $G_{2L-2}$  and edges in the stars of the spanner. We will prove that the resulting set of paths is  $\gamma$ -substantial (for certain  $\gamma$ ) in graph  $G$ . We will now describe both procedures SUBSTANTIAL-PATHS-IN-BLOCK and MAXIMAL-PATHS-IN-LAYERED-GRAPH. The analysis follows the description.

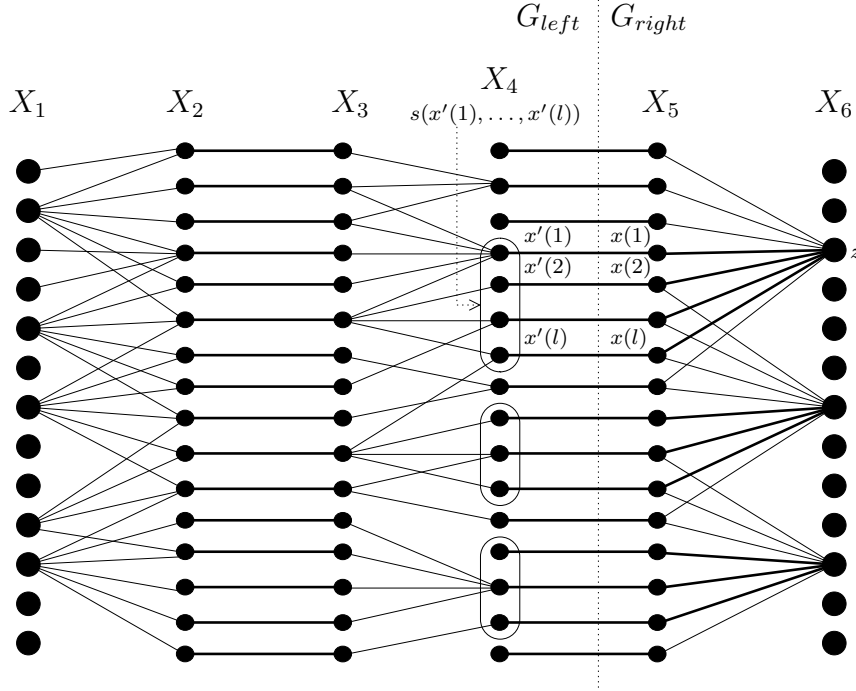


Figure 1: A  $(D_1, D_2)$ -block  $(X_1, X_2, \dots, X_{2L})$

---

PROCEDURE SUBSTANTIAL-PATHS-IN-BLOCK

*Input:* A  $(D_1, D_2)$ -block  $G = (X_1, \dots, X_{2L})$  with  $2L$  layers,  $L > 1$ .

*Output:* A substantial set of disjoint paths connecting  $X_1$  and  $X_{2L}$  in block  $G$ .

1. Find a  $(\frac{1}{2}, 16)$ -spanner  $S$  in  $G_{right}$ .
  2. Construct an auxiliary  $(2L - 2)$ -layered multi-graph  $G'_{left} = (X_1, \dots, X_{2L-3}, X')$  as follows:
    - For every star in the spanner  $S$  let  $x(1), \dots, x(l)$  be the vertices in  $X_{2L-1}$  that have degree one in  $S$  and let  $x'(1), \dots, x'(l)$  be the vertices in  $X_{2L-2}$  that are matched with  $x(1), \dots, x(l)$  by the matching  $M$  between  $X_{2L-2}$  and  $X_{2L-1}$ . Consider *super-vertex*  $s = s(x'(1), \dots, x'(l)) = \{x'(1), \dots, x'(l)\}$ . The set  $X'$  contains all super-vertices.
    - For every vertex  $x \in X_{2L-3}$  and every vertex  $y \in s(x'(1), \dots, x'(l))$ , if there is an edge between  $x$  and  $y$  in  $G_{2L-3}$  then put an edge between  $x$  and  $s$  in  $G'_{left}$ . Note that the number of edges between  $x$  and  $s$  is equal to the number of neighbors of  $x$  in  $s$ .
    - If edge  $e \in E(G_i)$  for some  $i \in \{1, \dots, 2L - 4\}$  then  $e$  is also in  $E(G'_{left})$ .
  3. Discard parallel edges in  $G'_{left}$ .
  4.
    - If  $L > 2$  then invoke MAXIMAL-PATHS-IN-LAYERED-GRAPH to find a maximal set of disjoint  $(M, 2L - 3)$ -paths in the graph obtained in step 3.
    - If  $L = 2$  then find a maximal matching (maximal set of disjoint  $(M, 1)$ -paths) in the graph from step 3 using the procedure from [HKP01].
  5. Extend paths found in step 4 to  $(M, 2L - 1)$ -paths in the block  $G$  using the edges of the matching in  $G_{2L-2}$  and the edges of the stars in spanner  $S$  from  $G_{right}$ .
-

First, let us address a small technical issue of SUBSTANTIAL-PATHS-IN-BLOCK. In the third step of the algorithm, parallel edges are discarded and then a maximal set of disjoint paths is found in the underlying simple graph. Since parallel edges only add repetitions of already present  $(M, 2L - 3)$ -paths, the set of paths found in step 4 which is maximal in the simple graph is also maximal in the multi-graph  $G'_{left}$ .

---

PROCEDURE MAXIMAL-PATHS-IN-LAYERED-GRAPH

*Input:* A layered graph  $G$  with  $2L$ -layers  $(X_1, \dots, X_{2L})$ .

*Output:* A maximal set of disjoint  $M$ -augmenting paths connecting  $X_1$  with  $X_{2L}$  in  $G$ .

1. For  $i := 1, \dots, (L - 1)\lceil \log n \rceil$  do:
  - For  $j := 1, \dots, \lceil \log n \rceil$  do:
    - (a) Let  $D_1 := \frac{n^{L-1}}{2^i}$  and  $D_2 := \frac{n}{2^j}$ .
    - (b) Iterate  $512L \log n$  times:
      - Invoke SUBSTANTIAL-PATHS-IN-BLOCK( $D_1, D_2$ ) in order to find a  $\gamma$ -substantial set of paths  $\mathcal{P}$  in the  $(D_1, D_2)$ -block and call MODIFY to modify graph  $G$  with respect to  $\mathcal{P}$ .

---

Next we analyze both procedures. First, we show that the set of paths obtained from SUBSTANTIAL-PATHS-IN-BLOCK is  $\gamma$ -substantial in  $G$  for some  $\gamma > 0$ . Let  $\mathcal{P}$  be the set of paths found by procedure SUBSTANTIAL-PATHS-IN-BLOCK,  $\mathcal{P}_{left}$  be the set of paths in  $G_{left}$  obtained from  $\mathcal{P}$  by restricting to  $G_{left}$ , and  $\mathcal{P}_{right}$  set of paths (of length one) in  $G_{right}$  obtained from  $\mathcal{P}$  by restricting to  $G_{right}$ .

**Lemma 3.1** *For every  $0 < \kappa < \frac{1}{4}$  either  $\mathcal{P}_{left}$  is  $(1 - 4\kappa)/4$ -substantial in  $G_{left}$  or  $\mathcal{P}_{right}$  is  $\kappa/16$ -substantial in  $G_{right}$ .*

**Proof** Let  $m = |X_{2L-1}| = |X_{2L-2}|$ . Let  $s_1, \dots, s_k$  be the super-vertices that are the endpoints of paths found in step 4. Every  $s_j$  contains some vertices from  $X_{2L-2}$ . In the last step of the procedure SUBSTANTIAL-PATHS-IN-BLOCK exactly one vertex from each  $s_j$  will be selected and each path will be extended using the edges of matching  $G_{2L-2}$  and edges of the spanner. Let  $x_j \in X_{2L-2}$  be the unique vertex from  $s_j$  which is selected and let

$$W = \bigcup_{j=1}^k (s_j \setminus \{x_j\}) \tag{2}$$

be the set of vertices contained in  $\bigcup_{j=1}^k s_j$  which are not selected. We distinguish two cases.

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

Since the spanner found in step 1 contains at least  $m/2$  vertices from  $X_{2L-1}$ , there are at least  $m/2$  vertices contained in super-vertices. Let  $\mathcal{Q}'$  denote the set of  $(M, 2L - 3)$ -paths between  $X_1$  and  $X'$  in  $G'_{left}$  and let  $\mathcal{Q}$  be the set of  $(M, 2L - 3)$ -paths between  $X_1$  and  $X_{2L-2}$  in  $G_{left}$ . We claim that

$$|\mathcal{Q}'| \geq \frac{1}{4} m D_1, \tag{3}$$

and

$$m D_1 \geq |\mathcal{Q}|. \tag{4}$$

To prove (3), note that there are at least  $m/2$  vertices contained in super-vertices and by definition of the  $(D_1, D_2)$ -block each such vertex is an endpoint of at least  $D_1/2$   $(M, 2L - 3)$ -paths between  $X_1$  and

$X_{2L-2}$ . Equation (4) follows from the definition of  $(D_1, D_2)$ -block. Combining (3) with (4) yields

$$|\mathcal{Q}'| \geq \frac{|\mathcal{Q}|}{4}. \quad (5)$$

The set of  $(M, 2L-3)$ -paths found in step 4 of the procedure is maximal in the maximal simple subgraph of  $G'_{left}$ . Consequently the set of  $(M, 2L-3)$ -paths is also maximal in the multigraph  $G'_{left}$ . Thus every path from  $\mathcal{Q}'$  shares a vertex with some path found in step 4 of the procedure. Also, every path from  $\mathcal{Q}'$  corresponds to a path from  $\mathcal{Q}$ . Indeed, path  $y_1 y_2 \dots y_{2L-3} s$  from  $\mathcal{Q}'$  contains vertices from layers  $X_1, \dots, X_{2L-3}$  and ends in a super-vertex  $s \in X'$ . Suppose that the super-vertex  $s$  contains  $x'(1), \dots, x'(l)$ . There are possibly a few edges between  $x'(1), \dots, x'(l)$  and  $y_{2L-3} \in X_{2L-3}$ . In step 5 exactly one neighbor of  $y_{2L-3}$  from  $x'(1), \dots, x'(l)$  is selected (arbitrarily) yielding the path in  $\mathcal{Q}$ . Note however that selecting only one vertex from  $s = \{x'(1), \dots, x'(l)\}$  can leave paths from  $\mathcal{Q}$  that end in other vertices from  $s$  to be disjoint with paths from  $\mathcal{P}_{left}$ . However in this case we have  $|W| < \kappa m$ , and so there are at most  $\kappa m D_1$  paths from  $\mathcal{Q}$  that end in  $W$ . Consequently the number of paths from  $\mathcal{Q}$  that share an endpoint with a path from  $\mathcal{P}_{left}$  is at least

$$|\mathcal{Q}'| - \kappa m D_1$$

which by (3) and (5) is at least

$$|\mathcal{Q}'|(1 - 4\kappa) \geq \frac{1 - 4\kappa}{4} |\mathcal{Q}|.$$

Therefore, the set of  $(M, 2L-3)$ -paths found by the procedure is  $(1 - 4\kappa)/4$ -substantial in  $G_{left}$ .

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

Let  $C$  denote the set of centers of stars in spanner  $S$  which are the endpoints of paths from  $\mathcal{P}$ . To show that  $\mathcal{P}_{right}$  is  $\kappa/16$ -substantial we will count the number of edges incident to vertices from  $C$ . We have

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

Consequently, the number of edges of  $G_{right}$  that share an endpoint with a path from  $\mathcal{P}_{right}$  is

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

and so  $\mathcal{P}_{right}$  is  $\kappa/16$ -substantial.  $\square$

An easy consequence of Lemma 3.1 is that the set of paths found by SUBSTANTIAL-PATHS-IN-BLOCK is  $\gamma$ -substantial in  $G$ .

**Lemma 3.2** *Let  $G$  be a  $(D_1, D_2)$ -block. If  $\mathcal{P}_{left}$  is  $\gamma$ -substantial in  $G_{left}$  or  $\mathcal{P}_{right}$  is  $\gamma$ -substantial in  $G_{right}$  then  $\mathcal{P}$  is  $\gamma/2$ -substantial in  $G$ .*

**Proof** Assume that  $\mathcal{P}_{left}$  is  $\gamma$ -substantial in  $G_{left}$ . Then the number of  $(M, 2L-1)$ -paths in  $G$  that are saturated by  $\mathcal{P}$  is at least

$$\gamma |\mathcal{Q}| \frac{D_2}{2}$$

where  $\mathcal{Q}$  is the set of  $(M, 2L-3)$ -paths in  $G_{left}$ . But the total number of  $(M, 2L-1)$ -paths in  $G$  is at most  $|\mathcal{Q}| D_2$  and so  $\mathcal{P}$  is  $\gamma/2$ -substantial.

Similarly if  $\mathcal{P}_{right}$  is  $\gamma$ -substantial in  $G_{right}$  then the number of  $(M, 2L-1)$ -paths in  $G$  that are saturated by  $\mathcal{P}$  is at least

$$\gamma |E(G_{right})| \frac{D_1}{2}.$$

But every vertex  $x \in X_{2L-2}$  is contained in at most  $D_1$   $(M, 2L-3)$ -paths in  $G_{left}$  and so the total number of  $(M, 2L-1)$ -paths in  $G$  is at most  $|E(G_{right})| D_1$ . Thus  $\mathcal{P}$  is  $\gamma/2$ -substantial.  $\square$

**Corollary 3.3** *The set of paths  $\mathcal{P}$  returned by PROCEDURE SUBSTANTIAL-PATHS-IN-BLOCK is  $1/256$ -substantial in  $G$ .*

**Proof** Apply Lemma 3.1 with  $\kappa = 1/8$  and then Lemma 3.2 with  $\gamma = 1/128$ . ▽

Next we analyze procedure MAXIMAL-PATHS-IN-LAYERED-GRAPH.

**Lemma 3.4** *Procedure MAXIMAL-PATHS-IN-LAYERED-GRAPH finds a maximal set of disjoint  $(M, 2L - 1)$ -paths in  $G$  in  $\log^{O(L)} n$  rounds.*

**Proof** First note that due to the modification done in each step of the procedure the resulting paths are disjoint. We shall show that the set of paths is also maximal. First, fix a  $(D_1, D_2)$ -block. By Corollary 3.3, SUBSTANTIAL-PATHS-IN-BLOCK( $D_1, D_2$ ) finds a  $1/256$ -substantial set of disjoint  $(M, 2L - 1)$ -paths in the  $(D_1, D_2)$ -block. Thus in the modification step, at least  $1/256$ -fraction of all  $(M, 2L - 1)$ -paths in the block will be deleted from it. However the total number of the  $(M, 2L - 1)$ -paths in  $G$  is at most  $n^{2L}$  and so after  $512 \cdot L \log n$  iterations, all  $(M, 2L - 1)$ -paths will be deleted from the block. In other words, the block will be empty. Note that there are at most  $L \log^2 n$  disjoint blocks, defined by parameters  $D_1, D_2$  in (a), and each  $(M, 2L - 1)$ -path of  $G$  belongs to exactly one of them. Thus after iterations in step 1 the set found by the procedure will be maximal in every block. Consequently the set of paths will be maximal in  $G$ . Finally let us estimate the running time of the procedure. First note that the order of the auxiliary graph is  $O(n)$  and each round in the auxiliary graph can be simulated in graph  $G$  in  $O(1)$  rounds. For  $i > 1$ , let  $R_i$  be the running time of the procedure invoked in the layered graph with  $2i$  layers and let  $R_1 = O(\log^4 n)$  be the time needed to find the maximal matching using the algorithm from [HKP01]. To estimate  $R_i$  notice that there are  $O(\log^2 n)$  iterations over all possible blocks and in each block  $O(\log n)$  iterations are done. In each iteration in a block SUBSTANTIAL-PATHS-IN-BLOCK( $D_1, D_2$ ) is invoked. This procedure finds a spanner in  $O(\log^3 n)$  steps (Lemma 2.6) and then invokes MAXIMAL-PATHS-IN-LAYERED-GRAPH in a layered graph with  $2i - 2$  layers (or finds a maximal matching in the bipartite graph when  $i = 2$ ). Thus,  $R_i = O(\log^3 n(\log^3 n + R_{i-1}))$ . Consequently,  $R_L = \log^{O(L)} n$ . ▽

## 4 Graphs without short odd cycles

In this section, we present our main procedure for computing a matching  $M$  from Theorem 1.1. As indicated before the main idea is to find a maximal matching  $M$  and then iterate with  $i = 1, \dots, k - 1$  and improve matching  $M$  by finding a maximal (with respect to inclusion) set of disjoint  $(M, 2i + 1)$ -paths in  $G$  and augmenting  $M$  using to obtain a larger matching. It is easy to see that once we augment  $M$  using a maximal set of disjoint  $(M, 2i + 1)$ -paths then the augmented matching  $M'$  will be such that there are no  $(M', 2i + 1)$ -paths in  $G$ . Finding a maximal set of disjoint  $(M, 2i + 1)$ -paths directly in graph  $G$  seems to be difficult. However, as we saw in the last section, if the graph has a layered structure, a maximal set of disjoint  $M$ -augmenting paths can be found by MAXIMAL-PATHS-IN-LAYERED-GRAPH. Building on this observation our strategy is to first reduce graph  $G$  and matching  $M$  to the auxiliary layered graph, then to find a maximal set of disjoint  $M$ -augmenting paths in the layered graph, and finally to translate the paths in the layered graph back to graph  $G$ . Unfortunately in the process of translation the maximality of the set of paths can be lost, i.e. even though the set of  $M$ -augmenting paths  $\mathcal{P}$  is maximal in the layered graph, the translated set  $\mathcal{P}'$ , obtained from  $\mathcal{P}$ , does not need to be maximal in  $G$ . However, as we will prove in this section,  $\mathcal{P}'$  will be  $\alpha$ -substantial in  $G$  for some  $\alpha > 0$  and so repeating the above three steps  $O(\log n)$  times will result in a maximal set of  $M$ -augmenting paths in  $G$ .

In what follows we will discuss three main procedures: REDUCE, TRANSLATE, and MAXIMAL-PATHS. Procedure REDUCE reduces graph  $G$  and matching  $M$  to the layered graph, procedure TRANSLATE obtains a set of  $M$ -augmenting paths in  $G$  given a maximal set of  $M$ -augmenting paths in the layered graph, and procedure MAXIMAL-PATHS contains the above algorithms and finds a maximal set of  $M$ -augmenting paths in  $G$ . Finally a complete algorithm is summarized in MAIN. Procedure TRANSLATE

requires three additional auxiliary procedures HEAVY-MIS, VERTEX-COLOR and CONST-MIS which are introduced just before TRANSLATE is discussed.

Procedure REDUCE constructs the layered graph  $Lay(G)$  from  $G$  and matching  $M$ .

---

PROCEDURE REDUCE

*Input:* Graph  $G$ , matching  $M$  in  $G$ , and number  $L$ .

*Output:* Layered graph  $Lay(G) = (X_1, \dots, X_{2L})$ .

1. Let  $U$  be the set of  $M$ -saturated vertices in  $G$  and let  $W$  be the set of  $M$ -free vertices in  $G$ .
  - Let  $X_1$  be a copy of  $W$  and let  $X_{2L}$  be another copy of  $W$ . Formally,  $X_1 := \{(w, 1) : w \in W\}$  and  $X_{2L} := \{(w, 2L) : w \in W\}$ .
  - For  $i = 2, \dots, 2L - 1$  let  $X_i$  be a copy of  $U$ . Formally let  $X_i := \{(u, i) : u \in U\}$ .
2. For  $k = 1, \dots, 2L - 1$  define  $G_k = Lay(G)[X_k, X_{k+1}]$  as follows.
  - If  $k$  is even then  $G_k := M$ , that is  $(u, k)$  is connected with  $(u', k+1)$  if and only if  $\{u, u'\} \in M$ .
  - If  $k$  is odd then  $(x, k)$  is connected with  $(x', k+1)$  if and only if  $\{x, x'\} \in E(G)$ .

---

Note that for every  $\{a_1, a_2\} \in M$ , both  $\{(a_1, i), (a_2, i+1)\}$  and  $\{(a_2, i), (a_1, i+1)\}$  will be  $G_{2i}$  for  $i = 1, \dots, L - 1$  (see Figure 1). There is a natural correspondence between  $(M, 2L - 1)$ -paths in  $Lay(G)$  and  $(M, 2L - 1)$ -paths in  $G$ .

**Definition 4.1** Let  $f$  be a function from the set of  $(M, 2L - 1)$ -paths in  $Lay(G)$  to the set of  $(M, 2L - 1)$ -paths in  $G$  given by

$$f(P) = x_1, x_2, \dots, x_{2L}$$

for an  $(M, 2L - 1)$ -path  $P = (x_1, 1), (x_2, 2), \dots, (x_{2L}, 2L)$  in  $Lay(G)$ .

Then  $f(P)$  starts in the  $M$ -free vertex  $x_1$ , ends in the  $M$ -free vertex  $x_{2L}$ , and by the definition of  $Lay(G)$ ,  $\{x_i, x_{i+1}\} \in E(G)$  and  $\{x_i, x_{i+1}\} \in M$  when  $i$  is even. Note however, that is not at all obvious that  $f(P)$  will be a path in  $G$  and it is the lack of short odd cycles which will let us prove in Lemma 4.2 that in fact all  $x_i$ 's are distinct and so  $f(P)$  is indeed an  $(M, 2L - 1)$ -path in  $G$ . We would like to mention two additional properties of  $f$  which will be important in the rest of the section. First let us note that trivially function  $f$  is surjective, i.e. for every  $(M, 2L - 1)$ -path  $Q$  in  $G$  there is an  $(M, 2L - 1)$ -path  $P$  in  $Lay(G)$  such that  $f(P) = Q$ . On the other hand  $f$  is not injective as  $f((x_1, 1), (x_2, 2), \dots, (x_{2L}, 2L)) = f((x_{2L}, 1), (x_{2L-1}, 2), \dots, (x_1, 2L))$ . We will often say that  $P$  and image of  $P$  by  $f$  correspond, i.e. if  $f(P) = Q$  then we will say that  $Q$  corresponds to path  $P$ .

**Lemma 4.2** Let  $G$  be a graph without odd cycles of lengths  $3, 5, \dots, 2L - 1$  and let  $M$  be such that there are no  $M$ -augmenting paths in  $G$  of lengths  $1, 3, 5, \dots, 2L - 3$ . Then for every  $(M, 2L - 1)$ -path  $P$  in  $Lay(G)$ ,  $f(P)$  is an  $(M, 2L - 1)$ -path in  $G$ .

**Proof** Let  $P = (x_1, 1), \dots, (x_{2L}, 2L)$  be an  $(M, 2L - 1)$ -path in  $Lay(G)$ . It is enough to show that all  $x_i$ 's are distinct. Note that  $x_1$  and  $x_{2L}$  are  $M$ -free in  $G$  and for  $i = 2, \dots, 2L - 1$ ,  $x_i$  is  $M$ -saturated in  $G$ . Consequently  $x_i \neq x_1$  and  $x_i \neq x_{2L}$  whenever  $2 \leq i \leq 2L - 1$ . Suppose that for some  $2 \leq i < j \leq 2L - 1$ ,  $x_i = x_j$ . Take such  $x_i, x_j$  with the smallest value of  $j - i$ . If  $j - i$  is odd then  $x_i x_{i+1} \dots x_j$  is an odd cycle in  $G$  of length  $j - i < 2L - 1$ . If  $j - i$  is even then there is a shorter  $M$ -augmenting path in graph  $G$ , as we can consider  $x_1 \dots x_{i-1} x_i x_{j+1} \dots x_{2L}$ . Since either both  $i, j$  are even or both are odd,  $x_1 \dots x_{i-1} x_i x_{j+1} \dots x_{2L}$  is indeed an  $M$ -augmenting walk of length smaller than  $2L - 1$ . The walk can be used to obtain either a shorter  $M$ -augmenting path or an odd cycle. Finally  $x_1$  and  $x_{2L}$  are different as otherwise  $G$  contains a cycle of length  $2L - 1$ .  $\nabla$

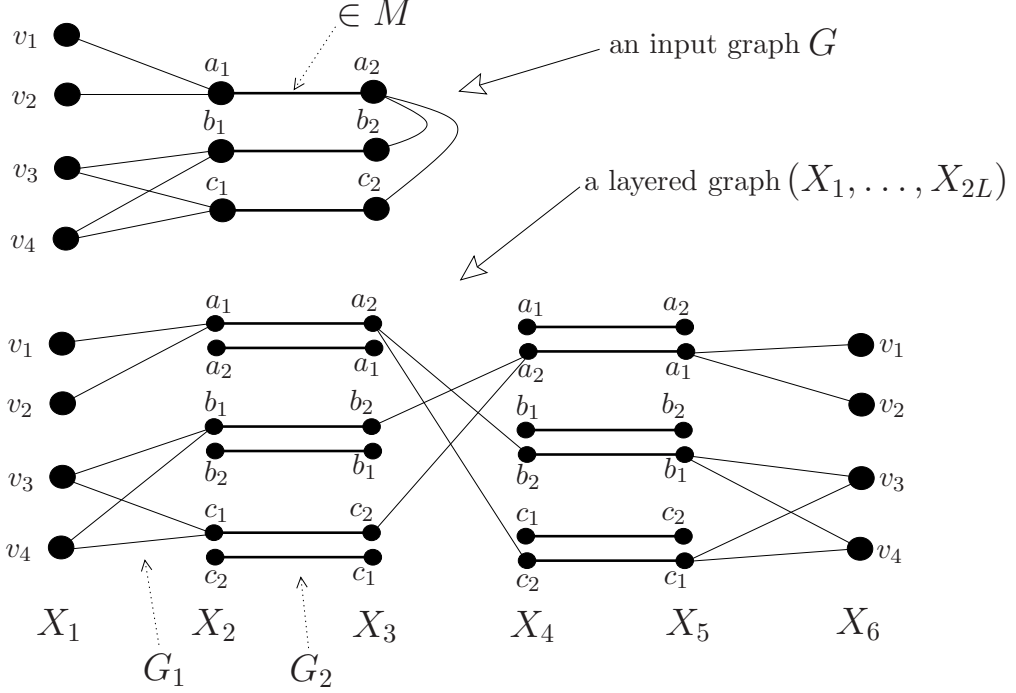


Figure 2: The reduction (augmenting paths of length 5).

Note that assumptions in Lemma 4.2 will be satisfied during the execution of our algorithm. First,  $G$  does not have odd cycles of lengths at most  $2k - 1$  by the assumption of Theorem 1.1 and the fact that  $L \leq k$ . Second,  $G$  will not have shorter  $M$ -augmenting paths as we will iterate with  $L = 2, 3, \dots, k$  and in the  $L$ th iteration, we shall eliminate all  $(M, 2L - 1)$ -paths. Finally, the initial matching  $M$  will be maximal and so there will be no  $(M, 1)$ -paths.

Let  $\mathcal{P}$  be a set of disjoint  $(M, 2L - 1)$ -paths in  $\text{Lay}(G)$  and let  $\mathcal{P}' = f(\mathcal{P})$  be the set of corresponding  $(M, 2L - 1)$ -paths in  $G$ . Define the *path graph*  $G_{\mathcal{P}}$  by  $V(G_{\mathcal{P}}) = \mathcal{P}$  and put an edge between two vertices  $P, P'$  of  $G_{\mathcal{P}}$  when the paths  $f(P)$  and  $f(P')$  intersect in  $G$ . Some computations will be done in  $G_{\mathcal{P}}$  and will require that the vertices of a graph have identifiers. We assume that the vertices of  $G$  have unique identifiers which are numbers bounded by a polynomial in  $|V(G)|$ . Define the identifiers of vertices of  $V(G_{\mathcal{P}})$  as follows. If  $\text{Id}(v)$  denote the identifier of vertex  $v$  in  $G$ , then for an  $(M, 2L - 1)$ -path  $P = (x_1, 1), (x_2, 2), \dots, (x_{2L}, 2L)$  in  $\text{Lay}(G)$  set  $\text{Id}(P) = [\text{Id}(x_1), \text{Id}(x_2), \dots, \text{Id}(x_{2L})]$  provided  $\text{Id}(x_1) < \text{Id}(x_{2L})$ . Note that  $\text{Id}(P)$  is a vector of length  $2L$  where each coordinate is an integer bounded by a polynomial of  $|V(G)|$ . It is convenient to write each of  $\text{Id}(x_i)$ 's in the binary form and think about  $\text{Id}(P)$  as a number consisting of concatenating binary expansions of  $\text{Id}(x_i)$ 's. Then,  $\text{Id}(P)$  is a number bounded by a polynomial in  $|V(G)|$ .

The main problem of using the  $\text{Lay}(G)$  is that even when paths from  $\mathcal{P}$  are disjoint in  $\text{Lay}(G)$  the corresponding paths do not need to be disjoint in  $G$ . Indeed, vertices in different layers of  $\text{Lay}(G)$  correspond to the same set of vertices in  $G$  (see Figure 4). However,  $G_{\mathcal{P}}$  cannot be completely arbitrary, in particular, as we will prove next, it must be a sparse graph of a constant (which depends on  $L$  only) maximum degree.

**Lemma 4.3** *Let  $\mathcal{P}$  be a set of paths obtained from MAXIMAL-PATHS-IN-LAYERED-GRAPH. Then the maximum degree of  $G_{\mathcal{P}}$ ,*

$$\Delta(G_{\mathcal{P}}) \leq 2 + (L - 1)(2(L - 1) - 1) \leq 2L^2 - 1.$$

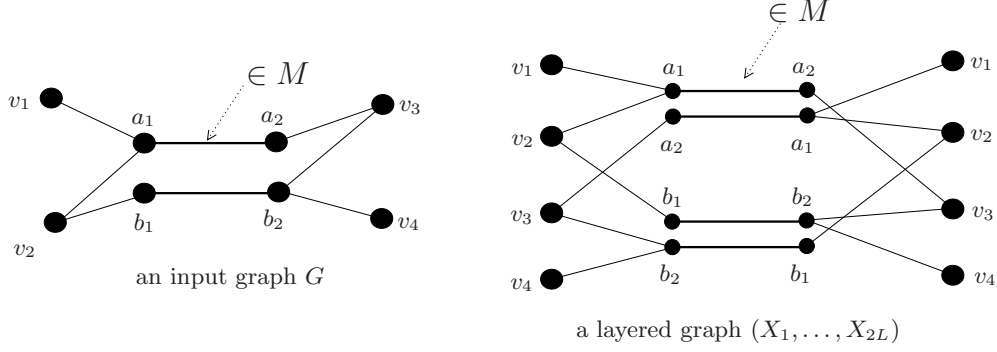


Figure 3: The reduction (augmenting paths of length 3).

**Proof** Let  $P = (x_1, 1), (x_2, 2), \dots, (x_{2L}, 2L)$  be a path from  $\mathcal{P}$ . Note that by definition of  $Lay(G)$ ,  $\{x_{2i}, x_{2i+1}\} \in M$  whenever  $i = 1, \dots, L-1$ . We will count the number of paths  $Q$  in  $\mathcal{P}$  such that  $f(P)$  and  $f(Q)$  intersect. Note that  $P$  and  $Q$  are disjoint in  $Lay(G)$  but they intersect in  $G$  and so  $f(P)$  and  $f(Q)$  either share an endpoint and are otherwise disjoint or, for some  $i$ , they share the edge  $\{x_{2i}, x_{2i+1}\}$  from  $M$ . If  $f(P)$  and  $f(Q)$  share an endpoint but are otherwise disjoint then as these are the endpoints are the only  $M$ -free vertices  $Q$  must either start in  $(x_{2L}, 1)$  or  $Q$  must end in  $(x_1, 2L)$  in  $Lay(G)$ . Consequently there can be at most two paths  $Q \in \mathcal{P}$  such that  $f(P)$  and  $f(Q)$  share an endpoint only. Now suppose that  $f(Q)$  contains the edge  $\{x_{2i}, x_{2i+1}\}$ . Then  $Q$  either contains  $\{(x_{2i}, 2j), (x_{2i+1}, 2j+1)\}$  or  $\{(x_{2i+1}, 2j), (x_{2i}, 2j+1)\}$  and for each  $j$  there can be only one path in  $\mathcal{P}$  with  $\{(x_{2i}, 2j), (x_{2i+1}, 2j+1)\}$  and one path with  $\{(x_{2i+1}, 2j), (x_{2i}, 2j+1)\}$ . As  $j = 1, \dots, L-1$  there are at most  $2(L-1)$  paths  $Q \in \mathcal{P}$  such that  $f(Q)$  contains  $\{x_{2i}, x_{2i+1}\}$  and one of them is  $P$ . Since  $i = 1, \dots, L-1$ , the degree of  $P$  in  $G_{\mathcal{P}}$  is at most  $2 + (L-1)(2(L-1) - 1)$ .  $\square$

Before we describe a procedure that translates disjoint paths in  $Lay(G)$  to disjoint paths in  $G$  we need to introduce the notion of an  $\alpha$ -heavy Maximal Independent Set (MIS). Let  $H = (W, F, weight)$  be a graph with weights on vertices, i.e.  $weight : W \rightarrow \mathbb{Z}^+$ . The weight of a subset  $A \subseteq W$  is then defined as  $weight(A) := \sum_{w \in A} weight(w)$ . Similarly the weight of a subgraph  $Q$  of  $F$  is defined as  $weight(Q) := weight(V(Q))$ . An  $\alpha$ -heavy MIS in  $H$  is a maximal independent set  $A$  of vertices such that  $weight(A) \geq \alpha weight(H)$ . Next, we show that if  $H$  has a constant maximum degree then for some  $0 < \alpha < 1$  an  $\alpha$ -heavy MIS can be found efficiently by a distributed algorithm. Let  $dist(v, A)$  denote the distance between  $v$  and  $A$ , i.e.  $dist(v, A) = \min_{a \in A} d(v, a)$  where  $d(v, a)$  is the length of the shortest path between  $v$  and  $a$ . Algorithm for an  $\alpha$ -heavy MIS uses an easy procedure CONST-MIS which finds a maximal independent set in a constant degree graph. Procedure CONST-MIS is described after the proof of Lemma 4.4 is given.

---

#### PROCEDURE HEAVY-MIS

*Input:* Graph  $G$  with maximum degree  $\Delta$  which is a constant independent of  $G$  and an upper bound  $maxweight$  ( $maxweight$  is a polynomial in  $|V(G)|$ ) for weight of vertices.

*Output:* An  $\alpha$ -heavy MIS,  $A$  in  $G$  with  $\alpha = \frac{1}{2(\Delta+1)^2}$ .

1. Let  $q := maxweight$  and  $A := \emptyset$ .
2. Let  $Q$  be a subgraph induced by all vertices  $v \in V(G)$  that satisfy  $weight(v) \in (q/2, q]$ .
3. Find a MIS,  $A'$  in  $Q$  using the trivial algorithm CONST-MIS.
4. Let  $A := A \cup A'$ . Remove from  $G$  all vertices  $v$  for which  $dist(v, A') \leq 1$ .

5.  $q := q/2$ ; if  $q \geq 1$  then go to step 2.

---

**Lemma 4.4** Procedure HEAVY-MIS finds an  $\frac{1}{2(\Delta+1)^2}$ -heavy MIS in a graph  $G$  with a constant maximum degree  $\Delta$  in  $O(\log^2 n)$  rounds.

**Proof** Let  $Q$  be an induced subgraph of  $G$  such that  $\forall v \in V(Q) : weight(v) \in (q/2, q]$  and let  $A'$  be a maximal independent set in  $Q$ . Every vertex from  $V(Q)$  has at least one neighbor in  $A'$  and so we have

$$weight(Q) \leq \sum_{a \in A'} q(1 + \Delta) = 2|A'| \frac{q}{2} (1 + \Delta) \leq 2weight(A')(1 + \Delta).$$

Consequently,

$$weight(A') \geq \frac{1}{2(\Delta + 1)} weight(Q). \quad (6)$$

Now let  $Q_i, A'_i$  be the sets  $Q$  and  $A'$  computed in the  $i$ th iteration of steps 2-4 of procedure HEAVY-MIS and let  $q_i = maxweight/2^i$  be the value of  $q$  in the same iteration. In the fourth step of the procedure we delete vertices from graph  $G$  which are within distance one (in  $G$ ) of  $A_i$  and so all vertices from  $V(Q_i)$  will be deleted as well as possibly some other vertices from  $V(G)$ . Let  $B_i$  be the set of vertices from  $V(G) \setminus V(Q_i)$  which are deleted in the fourth step of the procedure. Clearly,

$$\sum_{i=1}^k weight(V(Q_i) \cup B_i) = weight(G). \quad (7)$$

We also have

$$weight(Q_i) \geq \frac{1}{(1 + \Delta)} weight(V(Q_i) \cup B_i) \quad (8)$$

as  $weight(B_i) \leq \Delta \frac{q_i}{2} |V(Q_i)| \leq \Delta weight(Q_i)$ . From (6) and (8), we see that

$$weight(A'_i) \geq \frac{1}{2(\Delta + 1)} weight(Q_i) \geq \frac{1}{2(\Delta + 1)^2} weight(V(Q_i) \cup B_i)$$

and finally, by (7),

$$weight(A) = \sum_{i=1}^k weight(A'_i) \geq \frac{1}{2(\Delta + 1)^2} weight(G).$$

To complete the proof, note that the main loop has  $O(\log n)$  iterations because  $maxweight$  is a polynomial in  $n$ , and that MIS can be computed in  $O(\log n)$  rounds in a constant degree graph (see procedure CONST-MIS). Therefore the procedure HEAVY-MIS takes  $O(\log^2 n)$  rounds.  $\nabla$

To find a MIS in  $O(\log n)$  rounds in a constant degree graph  $G$  we first we find a proper  $(\Delta(G) + 1)$ -vertex coloring of  $G$ . Then we iterate over colors and in each step in parallel vertices from the  $i$ th color class decide if they join the independent set or not. Critically, vertices from a color class form an independent set and so the set will remain independent in this process. To find a proper  $(\Delta(G) + 1)$ -vertex coloring of  $G$  we invoke a well-know procedure which runs in  $O(\Delta \log n)$  rounds.

---

PROCEDURE VERTEX-COLOR( $V, i$ )

*Input:*  $G = (V, E)$ ; positive integer  $i$ .

*Output:* A proper  $(\Delta(G) + 1)$ -vertex-coloring of  $G$ .

1. If  $|V| = 1$  color vertex in  $V$  with color one.
2. If  $|V| > 1$  then:

- (a) Let  $Id_i(v)$  denote the  $i$ th bit of  $Id(v)$ . Partition  $V$  into two subsets  $V_0, V_1$  where  $V_j = \{v \in V, Id_i(v) = j\}$ .
- (b) Call recursively  $VERTEX-COLOR(V_0, i + 1)$  and  $VERTEX-COLOR(V_1, i + 1)$ .
- (c) For  $k = 1$  to  $\Delta + 1$  do:  
 In parallel for every vertex  $v$  in  $V_1$ : if  $v$  has color  $k$  then  $v$  colors itself with color  $c$  from the set  $\{1, \dots, \Delta + 1\}$  of colors so that every neighbor of  $v$  in  $V_0$  has color different than  $c$  connected.

To obtain a  $(\Delta + 1)$ -coloring of  $G = (V, E)$  we call  $VERTEX-COLOR(V, 1)$ . The number of rounds is clearly  $O(\log n)$  provided the identifiers are bounded by a polynomial in  $n$ . From the coloring it is easy to obtain a MIS. Procedure  $CONST-MIS$  finds a MIS in  $O(\log n)$  rounds in a constant degree graph  $G$  provided the identifiers of  $V(G)$  are bounded by a polynomial in  $n$ .

**PROCEDURE  $CONST-MIS$**

*Input:*  $G = (V, E)$  of a constant maximum degree  $\Delta$ .

*Output:* A maximal independent set  $X$  in  $G$ .

1. Call  $VERTEX-COLOR(V, 1)$  to find a proper  $(\Delta + 1)$ -vertex-coloring of  $G$ .
2. Let  $X$  be the set of all vertices colored in the first color.
3. For  $k = 2$  to  $\Delta + 1$  do:  
 In parallel, for every vertex  $v$  with color  $k$ : if  $v$  has no neighbors in  $X$  then add  $v$  to  $X$ .

Finally we can describe the translation procedure. The procedure will take as input set  $\mathcal{P}$  of disjoint  $(M, 2L - 1)$ -paths in  $Lay(G)$  and will return set  $\mathcal{P}'$  of disjoint  $(M, 2L - 1)$ -paths in  $G$ . Main property of the returned set  $\mathcal{P}'$  is that if  $\mathcal{P}$  is maximal in  $Lay(G)$  then  $\mathcal{P}'$  is  $\beta$ -substantial in  $G$  for some  $\beta > 0$ .

**PROCEDURE  $TRANSLATE$**

*Input:* Set  $\mathcal{P}$  of disjoint  $(M, 2L - 1)$ -paths in  $Lay(G)$ .

*Output:* Set  $\mathcal{P}'$  of disjoint  $(M, 2L - 1)$ -paths in  $G$ .

1. Consider the path-graph  $G_{\mathcal{P}}$ . For every vertex  $P$  in  $V(G_{\mathcal{P}})$  set  $weight(P)$  to be equal to the number of  $(M, 2L - 1)$ -paths in  $G$  that intersect path  $P$ .
2. Find an  $\alpha$ -heavy MIS,  $X$  in  $G_{\mathcal{P}}$  using  $HEAVY-MIS$ .
3. Set  $\mathcal{P}' := X$ .

**Theorem 4.5** *Let  $\mathcal{P}$  be a maximal set of disjoint  $(M - 2L - 1)$ -paths in  $Lay(G)$ . Then the set  $\mathcal{P}'$  returned by  $TRANSLATE$  is a  $\beta$ -substantial set of disjoint  $(M, 2L - 1)$ -paths in  $G$  with  $\beta = \frac{1}{16L^5}$ . In addition,  $TRANSLATE$  runs  $O(L \log^2 n)$  rounds.*

**Proof** First recall that two vertices from  $V(G_{\mathcal{P}})$  are connected in  $G_{\mathcal{P}}$  if and only if the corresponding paths in  $G$  share a vertex. Therefore an independent set in  $G_{\mathcal{P}}$  gives a set of disjoint paths in  $G$  and so, since  $\mathcal{P}'$  is an independent set in  $G_{\mathcal{P}}$ , the paths from  $\mathcal{P}'$  are disjoint. For an  $(M, 2L - 1)$ -path  $P$  in  $G$ , let  $weight(P)$  be the number of  $(M, 2L - 1)$ -paths in  $G$  (different than  $P$ ) that share a vertex with  $P$ . For a set  $\mathcal{Q}$  of  $(M, 2L - 1)$ -paths in  $G$  define

$$weight_G(\mathcal{Q}) := \sum_{P \in \mathcal{Q}} weight(P).$$

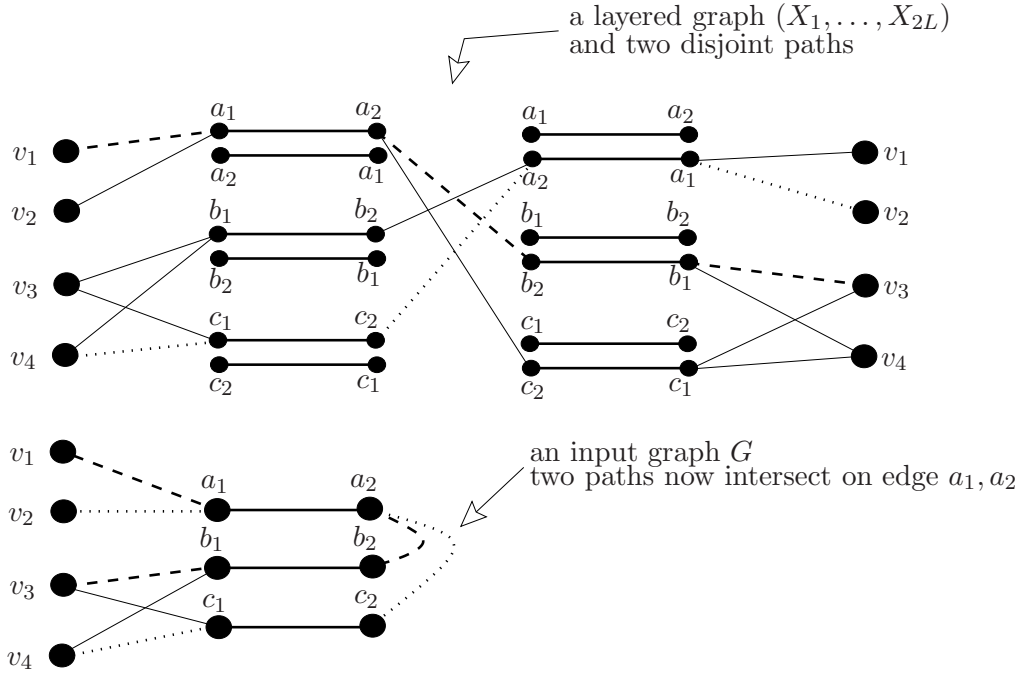


Figure 4: The translation (augmenting paths of length 5).

Note that  $weight_G(\mathcal{Q}) = weight_{G_P}(\mathcal{Q})$ . Let  $inc_G(\mathcal{Q})$  be the number of  $(M, 2L - 1)$ -paths in  $G$  that share a vertex with a path from  $\mathcal{Q}$ . First observe that  $weight_G(\mathcal{Q})$  and  $inc_G(\mathcal{Q})$  are quite related. In fact, the only difference between them is that if a path intersects more than one path from  $\mathcal{Q}$  then it will be counted once in  $inc_G(\mathcal{Q})$  and more than once in  $weight_G(\mathcal{Q})$ . Let  $\bar{\mathcal{P}}$  denote the set of not necessarily disjoint  $(M, 2L - 1)$ -paths in  $G$  that correspond to paths from  $\mathcal{P}$ . We have

$$weight_G(\bar{\mathcal{P}}) \geq inc_G(\bar{\mathcal{P}}). \quad (9)$$

Since every path from  $\mathcal{P}'$  contains  $2L$  vertices and the paths are disjoint, a path from  $G$  on  $2L$  vertices can intersect at most  $2L$  paths from  $\mathcal{P}'$ . As a result

$$inc_G(\mathcal{P}') \geq \frac{1}{2L} weight_G(\mathcal{P}'). \quad (10)$$

Therefore, by (9),(10), and Lemma 4.4, we have

$$inc_G(\mathcal{P}') \geq \frac{1}{4L(\Delta + 1)^2} inc_G(\bar{\mathcal{P}})$$

where  $\Delta$  is the maximum degree of  $G_P$ . By Lemma 4.3,

$$inc_G(\mathcal{P}') \geq \frac{1}{16L^5} inc_G(\bar{\mathcal{P}}).$$

To finish the proof, we will show that  $inc_G(\bar{\mathcal{P}})$  is equal to the total number of  $(M, 2L - 1)$ -paths in  $G$ . To that end, we argue that  $\bar{\mathcal{P}}$  is a maximal set of not necessarily disjoint  $(M, 2L - 1)$ -paths in  $G$ . It is enough to show that every  $(M, 2L - 1)$ -path in  $G$  shares a vertex with some path from  $\bar{\mathcal{P}}$ . Let  $\bar{P} = x_1, \dots, x_{2L}$  be a  $(M, 2L - 1)$ -path in  $G$ . Then the path  $P = (x_1, 1), \dots, (x_{2L}, 2L)$  is an  $(M, 2L - 1)$ -path in  $Lay(G)$ . Since  $\mathcal{P}$  is maximal in  $Lay(G)$ ,  $P$  must intersect some path  $Q = (y_1, 1), \dots, (y_{2L}, 2L)$

from  $\mathcal{P}$ , i.e. for some  $i$ ,  $y_i = x_i$ . Then however  $\bar{Q} = y_1 \dots y_{2L}$  is a path in  $\bar{\mathcal{P}}$  which intersects  $\bar{P}$ . Finally, observe that procedure TRANSLATE runs in  $O(L \log^2 n)$  rounds. Indeed, HEAVY-MIS in step 2 takes  $O(\log^2 n)$  rounds, and one round in the path-graph  $G_{\mathcal{P}}$  can be done in  $O(L)$  rounds in  $G$ .  $\nabla$

Next, we describe procedure MAXIMAL-PATHS which given an odd, positive integer, *shortest*, finds a maximal set of  $(M, \textit{shortest})$ -paths in  $G$ . In the final procedure MAIN we iterate with *shortest* =  $3, 5, \dots, 2k - 1$  and augment  $(M, \textit{shortest})$ -paths in each iteration.

---

PROCEDURE MAXIMAL-PATHS

*Input:* Graph  $G$  a matching  $M$  in  $G$  and a positive odd integer *shortest*.

*Output:* A maximal set of  $(M, \textit{shortest})$ -paths in  $G$ .

- Let  $\mathcal{P} := \emptyset$ . Let  $L := (\textit{shortest} + 1)/2$  and let  $b = 16L^5$ .
- Repeat  $((\textit{shortest} + 1) \cdot \log_b n)$  times:
  1. Call procedure REDUCE to construct the layered graph  $\textit{Lay}(G)$  with  $2L$  layers.
  2. Call MAXIMAL-PATHS-IN-LAYERED-GRAPH in  $\textit{Lay}(G)$  to find a maximal set of disjoint  $(M, 2L - 1)$ -paths in  $\textit{Lay}(G)$ .
  3. Call TRANSLATE to obtain a  $1/(16L^5)$ -substantial set  $\mathcal{P}'$  of disjoint  $(M, 2L - 1)$ -paths in  $G$ .
  4. Call MODIFY to modify the current graph with respect to  $\mathcal{P}'$ . Let  $\mathcal{P} := \mathcal{P} \cup \mathcal{P}'$ .
- Return the set  $\mathcal{P}$ .

---

**Theorem 4.6** *Let  $G$  be a graph without odd cycles of lengths  $3, 5, \dots, \textit{shortest}$  and let  $M$  be such that there are no  $M$ -augmenting paths in  $G$  of lengths  $1, 3, 5, \dots, \textit{shortest} - 2$ . Then procedure MAXIMAL-PATHS finds a maximal set of disjoint  $(M, \textit{shortest})$ -paths in  $G$ . In addition, MAXIMAL-PATHS runs in  $\log^{O(\textit{shortest})} n$  rounds.*

**Proof** Consider one iteration of MAXIMAL-PATHS. By Lemma 3.4, the set of paths found in the layered graph  $\textit{Lay}(G)$  is maximal. By Theorem 4.5, in the third step of the iteration MAXIMAL-PATHS finds a  $1/(16L^5)$ -substantial set  $\mathcal{P}'$  of disjoint  $(M, \textit{shortest})$ -paths in  $G$ . In the fourth step paths from  $\mathcal{P}'$  are added to  $\mathcal{P}$  and MODIFY is called to modify the graph with respect to  $\mathcal{P}'$ . As a result of the modification all of  $(M, \textit{shortest})$ -paths in  $G$  that share a vertex with a path from  $\mathcal{P}'$  are deleted. In addition, notice that paths added to  $\mathcal{P}$  in the  $i$ th iteration are disjoint from paths added to  $\mathcal{P}$  in the previous iterations. Consequently  $\mathcal{P}$  is the set of disjoint  $(M, \textit{shortest})$ -paths in  $G$ . Moreover, observe that  $\mathcal{P}$  is maximal, as in each iteration  $1/(16L^5)$ -fraction of the total number of  $(M, \textit{shortest})$ -paths in  $G$  is deleted. Since the total number of  $(M, \textit{shortest})$ -paths in a graph  $G$  on  $n$  vertices is at most  $n^{\textit{shortest}+1}$  after  $((\textit{shortest} + 1) \cdot \log_b n)$  iterations there will be no  $(M, \textit{shortest})$ -path in  $G$ . Consequently, every  $(M, \textit{shortest})$ -path in  $G$  will share a vertex with a path from  $\mathcal{P}$ .

Finally, there are  $O(\log n)$  iterations, each taking  $\log^{O(\textit{shortest})} n$  rounds.  $\nabla$

---

PROCEDURE MAIN

*Input:* An integer  $k \geq 1$ ; graph  $G$  without odd cycles of lengths less than or equal to  $2k - 1$ .

*Output:* A matching  $M$  in  $G$  such that there is no  $M$ -augmenting paths of length less than or equal to  $2k - 1$ .

1. Find a maximal matching  $M$  in  $G$  using the procedure from [HKP01].
2. For  $i := 2$  to  $k$  do:

- (a) Let  $shortest := 2i - 1$ . Call MAXIMAL-PATHS to find a maximal set  $\mathcal{P}$  of disjoint  $(M, shortest)$ -paths in  $G$ .
- (b) Augment all  $M$ -augmenting paths from  $\mathcal{P}$  and let  $M$  be the augmented matching.

---

Note that during the execution of the algorithm we improve the matching  $M$  computed in the first step. In particular as a result of augmenting paths from  $\mathcal{P}$  the edges from  $M$  will be substituted by the edges from  $E \setminus M$  that belong to paths from  $\mathcal{P}$ .

**Theorem 4.7** *Let  $k$  be an integer larger than 0 and let  $G$  be a graph on  $n$  vertices that does not have odd cycles of lengths less than or equal to  $2k - 1$ . Then procedure MAIN finds a matching  $M$  in  $G$  such that there are no  $M$ -augmenting paths in  $G$  of lengths less than or equal to  $2k - 1$ . In addition, MAIN runs in  $\log^{O(k)} n$  rounds.*

**Proof** After maximal matching  $M$  is found in the first step, MAIN iterates with  $shortest = 2i - 1$  where  $i = 2, \dots, k$  and invokes MAXIMAL-PATHS. Procedure MAXIMAL-PATHS finds a maximal set of disjoint  $(M, shortest)$ -paths in  $G$ . The paths are augmented in step 2(b) and so after this step, i.e. in the beginning of the next iteration, there will be no  $(M, shortest)$ -paths in  $G$ . Consequently assumptions of Theorem 4.6 are satisfied throughout the execution of MAIN. In particular, when the procedure exists there will be no  $M$ -augmenting path in  $G$  of length less than or equal to  $2k - 1$ . Finally, by Theorem 4.6, the running time of MAIN, is  $\log^{O(k)} n$ .  $\square$

**Proof of Theorem 1.1.** By Theorem 4.7, PROCEDURE MAIN finds a matching  $M$  so that there are no  $M$ -augmenting paths of lengths  $1, 3, 5, \dots, 2k - 1$ . Thus, by Theorem 1.2,  $|M| \geq \frac{k}{k+1} |M^*|$ .  $\square$

## References

- [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), 364-369.
- [Ch95] Z. Chen, A fast and efficient NC algorithm for maximal matching, Information Processing Letters, 55, (1995), 303-307.
- [CHS02] A. Czygrinow, M. Hańćkowiak, E. Szymańska, Distributed algorithm for approximating the maximum matching, Discrete Applied Mathematics, Volume 143, Issues 1-3, (2004), 62-71.
- [Ha95] Y. Han, An improvement on parallel computation of a maximal matching, Information Processing Letters, 56, (1995), 343-348.
- [HKP01] M. Hańćkowiak, M. Karoński, A. Panconesi, On the distributed complexity of computing maximal matchings, SIAM J. Discrete Mathematics, Vol. 15, No. 1, (2001), 41-57.
- [HK73] A. Hopcroft and R. Karp, An  $n^{5/2}$  algorithm for maximum matching in bipartite graphs, SIAM Journal on Computing, 2, (1973), 225-231.
- [Li92] N. Linial, Locality in distributed graph algorithms, SIAM Journal on Computing, 21(1), (1992), 193-201.
- [Lu86] M. Luby, A simple parallel algorithm for the maximal independent set problem, SIAM Journal on Computing, 15(4), (1986), 1036-1053.
- [Pe02] D. Peleg, *Distributed Computing. A Locality-Sensitive Approach*, SIAM Monographs on Discrete Mathematics and Applications, SIAM, 2000.