

Distributed algorithm for better approximation of the maximum matching ^{*}

A. Czygrinow¹ and M. Hańčkowiak²

¹ Department of Mathematics and Statistics
Arizona State University
Tempe, AZ 85287-1804, USA
andrzej@math.la.asu.edu

² Faculty of Mathematics and Computer Science
Adam Mickiewicz University, Poznań, Poland
mhanckow@main.amu.edu.pl

Abstract. Let G be a graph on n vertices that does not have odd cycles of lengths $3, \dots, 2k - 1$. We present an efficient distributed algorithm that finds in $O(\log^D n)$ steps ($D = D(k)$) matching M , such that $|M| \geq (1 - \alpha)|M^*|$, where M^* is a maximum matching in G , $\alpha = \frac{1}{k+1}$.

1 Introduction

We study the problem of finding an approximation to a maximum matching by a distributed algorithm. We consider the distributed network model 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 processors. The network is synchronized and in a single step each vertex can send messages to its neighbors, can receive the messages from its neighbors, and can perform some local computations. We make no explicit assumptions about the amount of local computations and the lengths of messages. The general problem in this setting is the following. Let G be a distributed network described above. Can vertices of G compute a global function of G in a relatively short time? In particular, the global function that we consider is a "large" matching, which we want to find in the poly-logarithmic (in $|V(G)|$) number of steps. Recent result of Hańčkowiak, Karoński, and Panconesi [HKP99] provides a breakthrough in this area. In [HKP99], authors presented an efficient distributed algorithm that finds a maximal matching. Note that, when M is a maximal matching, and M^* a maximum matching then $|M| \geq \frac{1}{2}|M^*|$. Using the techniques from [HKP99], Czygrinow, Hańčkowiak, and Szymańska [CHS02] designed an efficient distributed algorithm that finds a matching M such that $|M| \geq \frac{2}{3}|M^*|$. In this paper, we propose a distributed algorithm, that finds a better approximation of the maximum matching in case the graph does not have "short" odd cycles.

^{*} Research supported by KBN grant no. 7 T11C 032 20

Theorem 1. *Let k be an integer larger than one and let G be a graph on n vertices that does not have odd cycles of lengths $3, \dots, 2k - 1$. Then there is a constant $D = D(k)$ and distributed algorithm, that finds in $\log^D n$ steps matching M such that*

$$|M| \geq \frac{k}{k+1} |M^*|,$$

where M^* is a maximum matching in G .

If M is a matching in graph G , then 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 not M -saturated (see the next section for necessary definitions). Proof of Theorem 1 is based on the following fact (see [FGHP93]).

Theorem 2. *Let M^* be a maximum matching in graph G and let M be a matching such that there are no M -augmenting paths of lengths $1, 3, 5, \dots, 2k - 1$. Then*

$$|M| \geq \frac{k}{k+1} |M^*|.$$

In particular, the assumption that there are no M -augmenting paths of length one means that M is maximal. The algorithm from Theorem 1 proceeds as follows. First, we find a maximal matching M using the algorithm from [HKP99]. Then we iterate with $l = 2, \dots, k$ and "improve" matching M by finding a maximal set of M -augmenting paths of lengths $2l - 1$. Once the 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 does not have augmenting paths of lengths at most $2l - 1$. Consequently its size is at least $\frac{l}{l+1} |M^*|$. Finding the maximal set of M -augmenting paths is however not trivial and in fact it occupies the central part of the paper. The set is found by considering an auxiliary, virtual, graph constructed from G and matching M . It turns out that using, so-called spanners, a maximal set of M -augmenting paths can be found in this virtual graph (procedure MAXIMAL-PATHS-IN-LAYERED-GRAPH). Once found, it is translated to a maximal set of M -augmenting paths in the original graph G . To be able to claim that paths in the virtual graph correspond to paths (not cycles) in the original graph G the assumption about the cycle-structure of G is used. Finally, note that although there is a rich literature on distributed algorithms for matching problems most of the papers are focused on a different computational model than the one considered in this paper.

The rest of the paper is structured as follows. In the next section, we introduce necessary notation and auxiliary facts. Section 3 contains procedure that finds a maximal set of augmenting paths in the virtual 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 notation. Let M be a matching in 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.

Let P be a path in G and let M_1 and M_2 be two maximal matchings in P that partition the edges of P , i.e. $E(P) = M_1 \cup M_2$. We say that P is M -alternating if either $M_1 \subset M$ or $M_2 \subset M$. Path P of length $2k - 1$, $k \geq 1$, augments M if P is M -alternating and both ends of P are not M -saturated. These M -alternating paths that augment M will play a crucial role in our approach.

Definition 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 pairwise 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 graph $G = (V, E)$ if the number of M -saturated edges in G is at least $\gamma|E|$.

Using an algorithm from [HKP99] we can find a γ -substantial matching for any constant γ .

Lemma 1. Let n be the number of vertices of a graph. For any $\gamma > 0$ there is a distributed algorithm that finds in $O(\log^3 n)$ steps a γ -substantial matching.

The notion of a substantial matching extends in a natural way to a substantial set of paths.

Definition 3. 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 γ -path-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 $\gamma|\mathcal{M}|$.

Next, we define the **modification** of G with respect to a set of $(M, 2k - 1)$ -paths \mathcal{P} . The modification is composed of 2 steps:

- A) remove all edges of G that are incident to some path from \mathcal{P} ,
- B) remove all edges that do not belong to some augmenting paths of length $2k - 1$; ("new" edges of this sort can appear after step A).

Our algorithm uses the so called spanners in bipartite graphs. A bipartite 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 ingredient used in rendering a "substantial" set of disjoint paths will be a spanner.

Definition 4. An (α, β) -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 $b \in B$, $\deg_S(b) < \frac{\beta}{D}\deg_H(b) + 1$.

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 [HKP99]. In particular the following fact is proved in [HKP99].

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

One of the main parts of the algorithm will be a procedure that finds a set of disjoint paths in an auxiliary graph which has a layered structure. The number of layers in our auxiliary graphs will always be even and will always be a constant independent of n .

Definition 5. *Graph G is called a layered graph with $2L$ -layers if the vertex set of G can be partitioned into sets X_1, \dots, X_{2L} so that*

- $\forall i = 1..2L : X_i$ is an independent set.
- $\forall i = 1..2L - 1 : \text{vertices from } X_i \text{ are connected only with vertices from } X_{i+1}$.
- $\forall i = 1..L - 1 : |X_{2i}| = |X_{2i+1}|$ and the edges between X_{2i} and X_{2i+1} form a perfect matching.

We will often write $G = (X_1, X_2, \dots, X_{2L})$ to denote a $2L$ -layered graph with layers X_1, \dots, X_{2L} . Let G be as above, 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} . In particular, for even i , G_i is a perfect matching between X_i and X_{i+1} . In addition, let G_{left} be the graph induced by (X_1, \dots, X_{2L-2}) and, to make the notation uniform, let $G_{right} = G_{2L-1}$.

Definition 6. *Let $G = (X_1, X_2, \dots, X_{2L})$ be a layered graph and let $M = \bigcup_{i=1}^{L-1} E(G_{2i})$ be the matching in G . Then G is called a (D_1, D_2) -block if*

- $\forall v \in X_{2L-1} : d_{G_{2L-1}}(v) \in [\frac{D_2}{2}, D_2]$.
- $\forall v \in X_{2L-2} : \text{the number of } (M, 2L - 3)\text{-paths that have one endpoint in } X_1 \text{ another in } v \text{ is larger than } \frac{D_1}{2} \text{ and is at most } D_1$.

3 Maximal set of paths in a layered graph

In this section, we present an algorithm that finds a maximal set of augmenting paths in a layered graph. Layered graph will appear in a natural way as a virtual (auxiliary) graph in the course of the algorithm. The algorithm for finding a maximal set of augmenting paths consists of two main procedures: SUBSTANTIAL-PATHS-IN-BLOCK and MAXIMAL-PATHS-IN-LAYERED-GRAPH. The first one finds a set of disjoint 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. Procedure MAXIMAL-PATHS-IN-LAYERED-GRAPH iterates over all (D_1, D_2) -blocks for $D_1 = D_1(i) = \frac{n^{L-1}}{2^i}$, $D_2 = D_2(j) = \frac{n}{2^j}$. In each block it invokes SUBSTANTIAL-PATHS-IN-BLOCK

$O(\log n)$ times to find a maximal set of paths in the block. Since every augmenting path in the layered graph is a path of some block (and the blocks are disjoint) the union of the sets of paths will be maximal in the layered graph 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 that G_i is in fact a perfect matching between X_i and X_{i+1} when i is even. 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} . Each star (z, y_1, \dots, y_k) , with $z \in X_{2L}$ and $y_j \in X_{2L-1}$ is extended by considering the vertices $x_j \in X_{2L-2}$ that are matched with y_j by edges of matching G_{2L-2} . In the next step of the procedure new layered graph G'_{left} is constructed (in fact as explained later, G'_{left} is a multi-graph) with $2L-2$ layers $(X_1, X_2, \dots, X_{2L-3}, X')$ where X' contains so-called "super-vertices" that correspond to stars of the spanner. Then procedure calls MAXIMAL-PATHS-IN-LAYERED-GRAPH with G'_{left} as an input graph, and the paths obtained in G'_{left} are extended using the stars of the spanner. The resulting set of paths is substantial in graph G . More precisely we can describe the above procedure as follows.

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} . Create a *super-vertex* $s = s(x'(1), \dots, x'(l)) = \{x'(1), \dots, x'(l)\}$. The vertex set X' contains all super-vertices.
 - For every vertex $x \in X_{2L-3}$ and every $x'(j) \in s(x'(1), \dots, x'(l))$ put an edge between x and the super-vertex $s = s(x'(1), \dots, x'(l))$ in G'_{left} if x and $x'(j)$ are connected in G_{2L-3} .
 - Every edge of G that is present in G_i for some $i \in \{1, \dots, 2L-4\}$ is also present in G'_{left} .
3. Consider a maximal simple subgraph of G'_{left} (i.e. discard parallel edges).
 - If $L > 2$ then invoke MAXIMAL-PATHS-IN-LAYERED-GRAPH to find a maximal set of $(M, 2L-3)$ -paths in the subgraph of G'_{left} .
 - If $L = 2$ then find a maximal matching (maximal set of $(M, 1)$ -paths) in the subgraph using the procedure from [HKP99].
4. Extend paths found in (3) to $(M, 2L-1)$ -paths in the block G using the matching edges of G_{2L-2} and spanner S .

There is a small technical point that must be addressed at this moment. In the third step of the algorithm, parallel edges are discarded and then a maximal set of paths is found. Note that parallel edges only add the repetitions of $(M, 2L-3)$ -paths. Consequently the set of paths found in step 3 is also maximal in G'_{left} .

Next we shall show that the set of paths obtained from SUBSTANTIAL-PATHS-IN-BLOCK is substantial in G . Let \mathcal{P} be the set of paths found by the procedure, \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. *For every $0 < \kappa < \frac{1}{4}$ either \mathcal{P}_{left} is $(1 - 4\kappa)/4$ -path-substantial in G_{left} or \mathcal{P}_{right} is $\kappa/16$ -substantial in G_{right} .*

Proof of Lemma 3 is rather long and it mimics the proof of a similar fact from [CHS02]. It is an easy consequence of Lemma 3 that the set of paths found by SUBSTANTIAL-PATHS-IN-BLOCK is substantial in G .

Lemma 4. *If \mathcal{P}_{left} is γ -path-substantial in G_{left} or \mathcal{P}_{right} is γ -substantial in G_{right} then \mathcal{P} is $\gamma/2$ -path-substantial in G .*

We omit a routine proof.

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 augmenting paths connecting X_1 and 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) $D_1 := \frac{n^{L-1}}{2^j}$; $D_2 := \frac{n}{2^j}$;
 - (b) iterate $O(\log n)$ -times:
 - invoke SUBSTANTIAL-PATHS-IN-BLOCK(D_1, D_2) in order to find a substantial set of paths in the block and modify the graph G with respect to that set.

Lemma 5. *Procedure MAXIMAL-PATHS-IN-LAYERED-GRAPH finds a maximal set of disjoint paths in G in $\log^{O(L)} n$ steps.*

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 Lemma 4, SUBSTANTIAL-PATHS-IN-BLOCK(D_1, D_2) finds, for some fixed constant $\alpha > 0$, an α -path-substantial set of disjoint paths in a (D_1, D_2) -block. Thus in the modification step, at least α fraction of all paths in the block will be deleted from it. However the total number of the paths in whole G is at most $n^{O(L)}$ and so after $O(\log n)$ iterations all the paths will be deleted from the block. In other words, set of paths obtained after iterations in (b) will be maximal in the block. Note that there are $O(\log^2 n)$ disjoint blocks, defined by parameters D_1, D_2 in (a), and each 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 and consequently maximal in G . Finally let us estimate the running time of the procedure. First note that the order of the virtual graph is $O(n)$ and each step in the virtual graph can be simulated in graph G in

$O(1)$ steps. For $i > 1$, let R_i be the running time of the procedure invoked in the layered graph with $2i$ blocks and let $R_1 = O(\log^4 n)$ be the time needed to find the maximal matching using the algorithm from [HKP99]. 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 needed. 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) 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 odd cycles of length less than or equal to c .

In this section, we shall present our main procedure for computing a matching M from Theorem 1. As explained in the introduction our approach is based on finding M -augmenting paths. Finding a substantial set of disjoint M -augmenting paths in the input graph G seems to be difficult. However, as we saw in the last section, if a graph has a layered structure, augmenting paths can be found by MAXIMAL-PATHS-IN-LAYERED-GRAPH. The main problem that we must address is how to reduce graph G to a layered graph (Procedure REDUCE) and how to translate disjoint paths in the layered graph to the original graph G preserving the property that the set of paths is substantial (Procedure TRANSLATE). Next procedure constructs a virtual, layered graph from G .

PROCEDURE REDUCE

Input: a graph G , a matching M in G , a number L .

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

1. Lets denote by N vertices of G not saturated by M .
 $X_1 := N$; $X_i := \bigcup_{e \in M} e$, for $i = 2, \dots, 2L-1$; $X_{2L} := N$.
2. For $k = 1, \dots, 2L-1$ define $G_k = Lay(G)[X_k, X_{k+1}]$ as follows.
 For $i = 1, \dots, L$: $e \in G_{2i-1}$ if and only if there exist an augmenting path in G that has an edge e on the $2i-1$ position (counting from whatever end).
 For $i = 1, \dots, L-1$: $e \in G_{2i}$ if and only if $e \in M$.

Notice, that layers X_2, \dots, X_{2L-1} in $Lay(G)$ contain all vertices that are contained in edges of M . Thus every edge $\{a_1, a_2\} \in M$ appears twice in G_{2i} (see Figure 1). One important property of $Lay(G)$ is that every $(M, 2L-1)$ -path in G is present (has a corresponding $(M, 2L-1)$ -path) in $Lay(G)$. Using assumptions about G , we can show that the opposite direction is also true. Namely that every $(M, 2L-1)$ -path in $Lay(G)$ corresponds to a $(M, 2L-1)$ -path in G .

Lemma 6. *Let G be a graph without odd cycles of lengths $3, 5, \dots, 2L-1$ and let M be such that there are no augmenting paths in G of lengths $1, 3, 5, \dots, 2L-3$. Then every augmenting path connecting X_1 with X_{2L} in the layered graph corresponds to an augmenting path in graph G of length $2L-1$.*

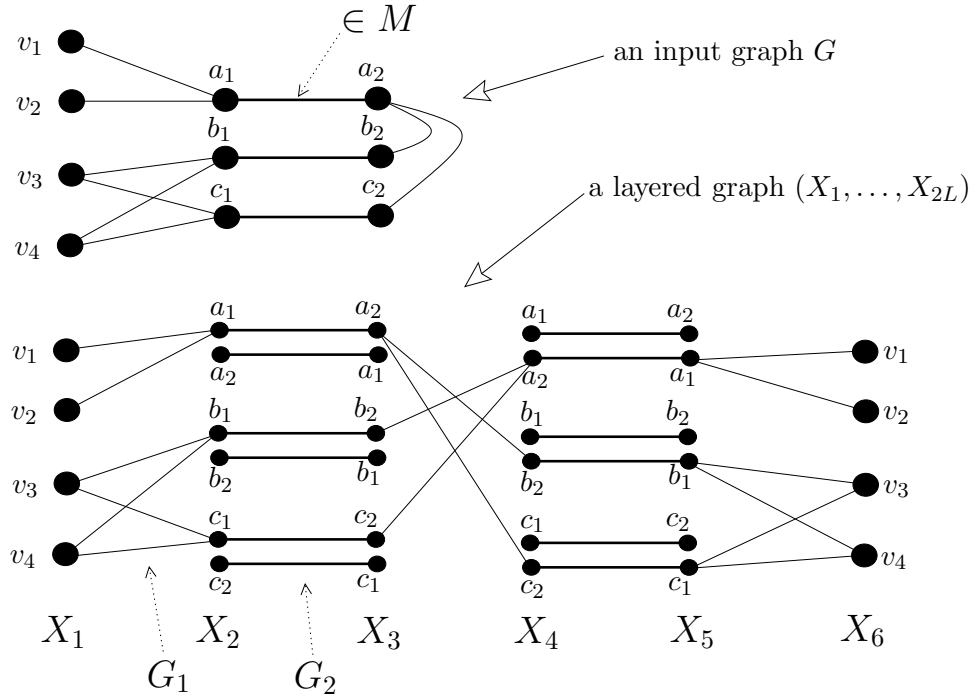


Fig. 1. The reduction (augmenting paths of length 5).

We omit the proof. Note that assumptions in Lemma 6 will be satisfied during the execution of our algorithm. First, G does not have odd cycles of lengths at most $2L - 1$ by the assumption. Second, G will not have shorter augmenting paths as we will iterate with $k = 3, 5, \dots, 2L - 1$ and in the k th iteration, we shall eliminate all (M, k) -augmenting paths. Finally, the initial matching M will be maximal and so no $(M, 1)$ -augmenting paths will be present. Let \mathcal{P} be a set of disjoint $(M, 2L - 1)$ -paths in $Lay(G)$. By Lemma 6 \mathcal{P} is also a set of $(M, 2L - 1)$ -paths in G . However the paths do not need to be disjoint in G . Define the **path graph** $G_{\mathcal{P}}$ by $V(G_{\mathcal{P}}) = \mathcal{P}$ and put an edge between two vertices of $G_{\mathcal{P}}$ when the corresponding paths intersect in G . In addition, we define the identifiers of vertices of $V(G_{\mathcal{P}})$. Let $Id(v)$ denote the identifier of vertex v in G . If $p = v_1, v_2, \dots, v_{2L}$ is a path in G which is a vertex in $V(G_{\mathcal{P}})$ and if $Id(v_1) < Id(v_{2L})$ then the identifier of p in $G_{\mathcal{P}}$ is the vector $[Id(v_1), Id(v_2), \dots, Id(v_{2L})]$. Although paths from \mathcal{P} don't need to be disjoint in G the maximum degree of $G_{\mathcal{P}}$ will be a constant that depends only on L .

Lemma 7. *Let \mathcal{P} be a set of paths obtained by MAXIMAL-PATHS-IN-LAYERED-GRAPH. The maximum degree of $G_{\mathcal{P}}$ is a constant independent of n .*

Before we describe the procedure that translates disjoint paths in $Lay(G)$ to disjoint paths in G we need to introduce the notion of a heavy Maximal Inde-

pendent Set (MIS). Let $H = (W, F)$ be a graph with weights on vertices, i.e. $w \in W$ has a weight $weight(w)$. The weight of a set of vertices A is then defined as $weight(A) := \sum_{w \in A} weight(w)$. Similarly the weight of a subgraph Q is defined as $weight(Q) := weight(V(Q))$. The α -heavy MIS in H is a maximal independent set A of vertices such that $weight(A) \geq \alpha weight(H)$. We note here, that for a constant d a $\frac{1}{2(d+1)^2}$ -heavy MIS can be found by a distributed algorithm in a graph G with $\Delta(G) = d$, in time $O(\log^2 n)$. Details of this procedure will appear in the journal version of the paper.

Now we can describe procedure TRANSLATE:

PROCEDURE TRANSLATE

Input: a set \mathcal{P} of disjoint, augmenting paths connecting sets X_1 and X_{2L} in the layered graph; \mathcal{P} is maximal in the layered graph.

Output: a set \mathcal{P}' of disjoint, augmenting paths of lengths $2L - 1$ in the input graph G ; \mathcal{P}' is path-substantial in G .

1. For each vertex v of the path graph $G_{\mathcal{P}}$ assign the value $weight(v)$ equal to the number of augmenting paths of length $2L - 1$ in G that are incident to v (or in other words "touch" v).
2. Compute a heavy MIS, X in $G_{\mathcal{P}}$.
3. The set \mathcal{P}' contains paths that correspond to vertices of X .

Theorem 3. *Let G be as in Lemma 6 and let $Lay(G)$ be the layered graph with $2L$ layers obtained from G by procedure REDUCE. Let \mathcal{P} be the set of paths in $Lay(G)$ obtained by MAXIMAL-PATHS-IN-LAYERED-GRAPH. Then the set \mathcal{P}' obtained from \mathcal{P} by TRANSLATE is a β -path-substantial set of disjoint paths in G for certain constant β . In addition, TRANSLATE runs $O(L \log^2 n)$ steps.*

We omit the proof. Next, we describe the procedure MAXIMAL-PATHS that finds a maximal, set of $(M, shortest)$ -paths in G for an odd, positive integer, $shortest$. Note that G and M are as in Lemma 6 with $shortest = 2L - 1$. In the final procedure MAIN we iterate with $shortest = 3, 5, \dots, c$ to obtain a maximal set of (M, c) -paths in G .

PROCEDURE MAXIMAL-PATHS

Input: an input graph G ; a matching M ; a number $shortest$.

Output: a maximal, set of $(M, shortest)$ -paths in G .

- Let $\mathcal{P} := \emptyset$.
- Repeat $O(\log n)$ times:
 1. Call procedure REDUCE to build a layered graph with $2L$ -layers (X_1, \dots, X_{2L}) , where $L := (shortest + 1)/2$.
 2. Call MAXIMAL-PATHS-IN-LAYERED-GRAPH on layered graph to find maximal set of disjoint, augmenting paths connecting sets X_1 and X_{2L} .
 3. Call TRANSLATE to get a path-substantial set \mathcal{P}' of disjoint, augmenting paths in G .
 4. Modify the current graph with respect to \mathcal{P}' . Let $\mathcal{P} := \mathcal{P} \cup \mathcal{P}'$

- The set \mathcal{P} is the result of that procedure.

Theorem 4. *Let G be a graph without odd cycles of lengths $3, 5, \dots$, shortest and let M be such that there are no M -augmenting paths in G of lengths at most $\text{shortest} - 2$. Then procedure MAXIMAL-PATHS finds a maximal set of disjoint $(M, \text{shortest})$ -paths in G . In addition, MAXIMAL-PATHS runs in $\log^{O(\text{shortest})} n$ steps.*

PROCEDURE MAIN

Input: an odd number $c \geq 3$; graph G without odd cycles of length less than or equal to c .

Output: a matching in G such that there is no augmenting path of length less than or equal to c .

1. Compute a maximal matching in G using the procedure from [HKP99].
2. For $\text{shortest} := 3$ to c step 2 do:
 - (a) Call MAXIMAL-PATHS to find a maximal set of disjoint, augmenting paths of length shortest .
 - (b) Augment all augmenting paths.

Note that during the execution of the algorithm we try to "improve" the matching computed in step 1: some of the edges of the matching will be deleted, and some will be added to it as we augment paths in 2(b). We can summarize the discussion in the following theorem.

Theorem 5. *Let c be an integer larger than two and let G be a graph on n vertices that does not have odd cycles of lengths less than or equal to c . 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 c . In addition, MAIN runs in $\log^{O(c)} n$ steps.*

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

References

- [CHS02] A. Czygrinow, M. Hańkowiak, E. Szymańska, *Distributed algorithm for approximating the maximum matching*, submitted, 2002.
- [FGHP93] T. Fischer, A. V. Goldberg, D. J. Haglin, S. Plotkin, *Approximating matchings in parallel*, Information Processing Letters, 1993, 46, pp. 115-118.
- [HKP99] M. Hańkowiak, M. Karoński, A. Panconesi, *A faster distributed algorithm for computing maximal matching deterministically*, Proceedings of PODC 99, the Eighteen Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, pp. 219-228.
- [Li92] N. Linial, *Locality in distributed graph algorithms*, SIAM Journal on Computing, 1992, 21(1), pp. 193-201.