

Distributed Packing in Planar Graphs

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

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

Wojciech Wawrzyniak†
Faculty of Mathematics and
Computer Science
Adam Mickiewicz University
Poznań, Poland
wwawrzy@amu.edu.pl

ABSTRACT

We give an efficient distributed algorithm that finds an almost optimal packing of a graph H in a planar graph G . The algorithm is deterministic and its running time is poly-logarithmic in the order of G .

Categories and Subject Descriptors

F.2.8 [Analysis of algorithms and problem complexity]: Non-numerical algorithms-Computation of discrete structures

General Terms

Algorithms, Theory

Keywords

Distributed algorithms, approximation algorithms, graph packing

1. INTRODUCTION

Finding distributed complexity of classical graph theoretic problems in the message-passing model is a challenging and vastly open area of research. In the case of general graphs, efficient (the number of rounds is poly-logarithmic in the order of a graph) deterministic distributed algorithms are difficult to design. In fact among classical graph-theoretic problems only the distributed complexity of the maximal matching problem is well understood in the model considered

*This work was supported by grant N206 017 32/2452 for years 2007-2010

†This work was supported by grant N206 017 32/2452 for years 2007-2010

in this paper and the problem is known to admit an efficient solution [10]. At the same time even the distributed complexity of the maximal independent set problem is still an outstanding open problem. On the other hand there has been a modest success in determining the complexity of problems in the case when the underlying network has special properties. For example, in the case of constant degree graphs, unit-disk graphs, or planar graphs many problems which seem to be unapproachable for general networks, admit efficient and often simple distributed algorithms. Certainly, the most striking example of how the special properties of the graph can be exploited is the algorithm of Cole and Vishkin from [2] that gives a very fast (the number of rounds is $O(\log^* |G|)$) deterministic solution to the Maximal Independent Set (MIS) problem in constant-degree graphs. In unit-disk graphs, an efficient deterministic algorithm for the MIS problem is given in [11] and in the case of planar graphs a $O(\log |G|)$ -time algorithm is very easy to design. In fact, in the case of special classes of graphs, much more can be usually accomplished for a very similar price as the one paid to find MIS. In particular, in unit-disk graphs it is possible to find a very good approximation of a maximum independent set, as shown in [12], and related problems like the maximum matching, the minimum dominating set, or the minimum-connected dominating set problem admit efficient approximations as well (see [12] and [4]). Even the packing problem is known to have an efficient distributed solution in the unit-disk graphs (see [6]). The situation is similar in planar graphs or more generally in the case of minor-closed families of graphs where efficient approximations for the maximum independent set problem, the maximum matching, and the dominating set problems are known ([3]). In this paper we study the distributed complexity of the packing problem in planar graphs and show that it is possible to efficiently approximate the packing number of a graph H in a planar graph G provided the order of H is poly-logarithmic in the order of G . The algorithm on one hand generalizes the approximation procedure for the maximum matching prob-

lem from [7] and on the other, complements the result from [6] and shows that the family of planar graphs is another class where the packing problem can be approximated efficiently. Maybe, it is interesting to note that the algorithm described in the current paper differs in many aspects from algorithms for minor-closed families from [3]. For example, algorithms from [3] only need the fact that subgraphs (that arise during iterations of the main procedure) which are minors of the underlying network G exhibit similar properties to the ones of G . In the case of the packing problem, the analysis of the algorithm relies heavily on the fact that G is planar and it is not possible to extend it to minor-closed families of graphs. Finally, we note that although the only assumption needed in our packing algorithm is that G is planar, the analysis of the procedure uses properties of a plane drawing of G .

1.1 Terminology and notation

We consider the message-passing distributed model as formalized for example by Linial in [13]. In this model the underlying network forms an undirected graph with vertices corresponding to computational units, and edges corresponding to communication links between them. The network is synchronized and computations proceed in discrete rounds. In one round each vertex can send and receive messages from its neighbors, and can perform some local computations. Neither the amount of local computations nor the lengths of messages are restricted in any way. We will also assume that nodes in the network have unique identifiers from $\{1, \dots, |G|\}$ where $|G|$ is the order of G and is globally known.

Following [13] we call a distributed algorithm *efficient* if it runs in a poly-logarithmic (in the order of the graph) number of rounds. We will use the graph-theoretic notation and terminology from [8] and in particular we use $|G|$ to denote the order of G and $\|G\|$ to denote the size of G . Finally recall some of the terminology concerning the packing problem. Let H and G be graphs. A family of graphs $\{H_1, \dots, H_l\}$ is called an l -packing of H in G if for every $i \in \{1, \dots, l\}$, H_i is isomorphic to H , H_i is a subgraph of G and whenever $i \neq j$, $V(H_i) \cap V(H_j) = \emptyset$. The packing number of H in G , denoted by $\nu_H(G)$, is the largest l such that G admits an l -packing of H . For example the packing number $\nu_{K_2}(G)$ is the size of the maximum matching in G .

1.2 Results and related work

For an overview of distributed approximation algorithms we refer the reader to the survey by Elkin [9] and for a comprehensive guide to the theory of distributed algorithms to [14]. As mentioned before we study the distributed approximation algorithm for the packing problem in planar graphs and our algorithm gives a generalization of the maximum matching ap-

proximation from [7] as well as an analog of the result from [6] in the world of planar graphs.

The main result of this paper is a distributed deterministic algorithm which finds in a planar graph G an l -packing of graph H with l which is approximately equal to $\nu_H(G)$. More precisely, we give a distributed algorithm which given a positive integer k and a graph H finds in a planar graph G an l -packing of H with $l \geq (1 - 1/\log^k |G|)\nu_H(G)$. The algorithm runs in a poly-logarithmic (in $|G|$) number of rounds provided $|H| = \text{polylog}(|G|)$ and $k = O(1)$. In addition, we observe that a slightly more general problem of packing allowable subgraphs (see Section 3.2 for the precise definition) can be solved using the same algorithm.

The main idea behind our approach is as follows. First find in a planar graph G a special partition of $V(G)$ and a small set $Z \subset V(G)$ which is a vertex cover of the cross-edges of the partition. Note that finding such a set Z is far from trivial and we give a general idea of how this is accomplished in the beginning of the next section. After Z and the partition are obtained, contract the partition classes to single vertices and use the set Z to define the weights of the contracted vertices. Next find another partition of the weighted graph obtained after contraction using the procedure from [5]. This new partition is such that the weight of the border vertices is negligible with respect to the weight of the whole graph which by a simple trick can be shown to be at most the packing number of H in G . Finally find an optimal packing in each of the graphs induced by partition classes and return the union. Intuitively, the part of an optimal solution that contains cross-edges can be bounded by the size of the maximum matching in the subgraph induced by edges between distinct partition classes which in turn can be bounded by the size of a minimum vertex cover. Consequently the fact that the weight of the cover is negligible allows one to conclude that the returned solution is a good approximation. Note however that finding an appropriate partition of $V(G)$ and the set Z is non-trivial and an algorithm for accomplishing just one step in that direction (the separation procedure from the next section) constitutes a substantial ingredient of our approximation algorithm.

1.3 Organization

The rest of the paper is structured as follows. First we give a separation procedure that can be viewed as a single step of the algorithm for finding the special partition and the cover. Next, in Section 3, we use the separation algorithm to give the partitioning procedure and concoct ingredients to obtain a distributed approximation algorithm for the packing problem.

2. BUNCHES AND SEPARATION

In this section, we give a separation procedure. The starting point of the algorithm SEPARATION that we describe in this section is a very special partition of a planar graph G . The objective of the procedure is to find a "small" set of vertices in the graph that separates each partition class from the rest of the graph. It is important to note that when invoked (in CLUSTERS step (c), Section 3.1) the procedure works in a graph obtained by extensive preprocessing. In particular two partitions of $V(G)$, \mathcal{P} and \mathcal{C} , with \mathcal{C} refining \mathcal{P} are given and the main objective of the separation algorithm is to separate interiors of partition classes of \mathcal{P} (see Lemma 2.11) using the number of vertices which is proportional to $|\mathcal{C}|$ (see Lemma 2.10). Before we proceed let us recall some graph-theoretic terminology (we follow [8] in most of the discussion). A graph will always mean a simple graph and a multigraph will mean a multigraph in which multiple edges are allowed but loops are not. If F is a tree and u, v are vertices of F then uFv denotes the unique path in F that starts in u and ends in v . In a general graph F , a path connecting u, v in F will be called a $u-v$ path and in the case of multigraph an edge joining u, v will be called a $u-v$ edge. For two vertices u, v in G the distance between u and v , denoted by $d_G(u, v)$, is the length of the shortest $u-v$ path and the diameter of G is the maximum distance. If H is a subgraph of G then the diameter of H is simply the diameter of the subgraph H and the weak diameter of H is the maximum distance in G between two vertices from H . In the next subsection, we introduce a few additional concepts (in particular, the notion of a bunch is critical for the analysis of SEPARATION) and establish some properties of them. Then, in 2.2, give SEPARATION and analyze it.

2.1 Bunches

Let us start with a definition of an l -dominating set.

DEFINITION 2.1. *Let l be a positive integer and let $G = (V, E)$ be a graph. A subset D of V is called an l -dominating set in G if for every vertex $v \in V$ there is a vertex $u \in D$ such that $d_G(u, v) \leq l$.*

Let D be an l -dominating set in $G = (V, E)$ and let $\mathcal{P} = (P_1, \dots, P_k)$, $\mathcal{C} = (C_1, \dots, C_l)$ be two partitions of V such that

- (i) \mathcal{C} is a refinement of \mathcal{P} , that is every C_i is a subset of some P_j ,
- (ii) every C_i contains exactly one vertex from D (denoted by d_i), and $G[C_i]$ is a connected subgraph of G .

In addition we assume that for every $i = 1, \dots, l$ there is a (spanning) breadth-first search tree T_i in $G[C_i]$

which is rooted at d_i . Sets C_i will be called *small clusters* and sets P_j will be called *big clusters*.

DEFINITION 2.2. *Let $d_i, d_j \in D$. A d_i - d_j path is called special if it has the form $d_i P u v P' d_j$ for some $u \in C_i, v \in C_j$ where P is the path in T_i that connects d_i with u and P' is the path in T_j that connects v with d_j .*

We note that the number of special paths between d_i and d_j is equal to the number of edges between C_i and C_j . Indeed, for every edge uv with $u \in C_i$ and $v \in C_j$ there is a unique d_i-u path in T_i and a unique $v-d_j$ path in T_j .

Although procedure SEPARATION which we describe in Section 2.2 works in planar graphs, in the analysis of its correctness we will assume that a given graph is plane and we will use properties of the plane drawing. Before continuing, let us recall some basic graph-theoretic terminology relating to plane graphs. If G is a plane graph in R^2 then a maximal open set f in $R^2 \setminus G$ such that any two point in f can be connected by a curve contained in f is called a face of G . Let P, Q be two special d_i-d_j paths. In any plane drawing, graph $P \cup Q$ contains exactly one bounded face. (We will assume here that the face is empty if $P = Q$.) Now we set $F(P \cup Q) := f$ and $Reg[P \cup Q] := (P \cup Q) \cup f$ where f is the bounded face in the drawing of $P \cup Q$.

DEFINITION 2.3. *Let $G = (V, E)$ be a plane graph and let $d_i, d_j \in D$ with $i \neq j$. A maximal set \mathcal{B} of special paths between d_i and d_j will be called a d_i-d_j bunch if there exist two (not necessarily distinct) paths $P, Q \in \mathcal{B}$ such that all paths from \mathcal{B} are contained in $Reg[P \cup Q]$ and such that no vertex from D is contained in $F(P \cup Q)$. In addition, the paths P, Q will be called the boundary paths of \mathcal{B} .*

Note that there can be more than one d_i-d_j bunch but in that case bunches have the following property.

FACT 2.4. *Let G be a plane graph, let $d_i, d_j \in D$ with $i \neq j$, and let \mathcal{A} and \mathcal{B} be two d_i-d_j bunches. If $\mathcal{A} \neq \mathcal{B}$ then $\mathcal{A} \cap \mathcal{B} = \emptyset$ and for every path $P \in \mathcal{A}$ and every path $Q \in \mathcal{B}$ there is a vertex from D contained in $F(P \cup Q)$.*

Proof. Assume that $\mathcal{A} \neq \mathcal{B}$. Then $\mathcal{A} \cap \mathcal{B} = \emptyset$ as otherwise by maximality $\mathcal{A} = \mathcal{B}$. If $P \in \mathcal{A}$ and $Q \in \mathcal{B}$ then as $P \neq Q$, the face $F(P \cup Q)$ is non-empty and if there was no vertex from D in $F(P \cup Q)$ then P and Q would be in the same bunch. \square

Let \mathbf{B} be the set of all bunches and let M_G be the multigraph obtained from G by contracting each C_i to a vertex and adding an edge between contracted vertices c_i, c_j for every $d_i - d_j$ bunch. Then the number of edges in M_G between c_i and c_j is equal to the number of $d_i - d_j$ bunches. Consequently

$$|\mathbf{B}| = \|M_G\|. \quad (1)$$

In addition, M_G is a planar multigraph and we can assume that M_G is a plane multigraph vertices of which have the same positions in R^2 as vertices from D in the plane drawing of G . Then the following analog of Fact 2.4 holds.

FACT 2.5. *Let e, e' be two $u - v$ edges in M_G . If $e \neq e'$ then there is a vertex from $V(M_G)$ contained in the bounded face of the cycle $ueve'$.*

As we will show next, the number of edges in M_G can be bounded using Euler's formula.

FACT 2.6. *Let G be a connected plane graph and let $|D| \geq 2$. Then $|M_G| = |D|$ and $\|M_G\| \leq 18|D| - 24$.*

Proof. We clearly have $|M_G| = |D|$. Since G is connected so is M_G and thus it contains a spanning tree T . Consider a plane drawing of M_G and for every vertex $v \in V(T)$ let $\epsilon_v > 0$ be such that a ball B_v around v of radius ϵ_v intersects only with those edges of M_G that contain v and does not contain points from other balls. Call a connected region of $B_v \setminus T \subseteq R^2$ a *side* of v . Every edge from $E(M_G) \setminus E(T)$ that contains v reaches v by some side s of v . In this case we will say the edge ends in s .

CLAIM 2.7. *Let u, v be two vertices. There can be at most two $u - v$ edges e, e' of $E(M_G) \setminus E(T)$ such that e and e' end in the same side of u and the same side of v .*

Proof of Claim 2.7. Let F be the set of $u - v$ edges from $E(M_G) \setminus E(T)$ that end in the same sides of u and v . For $e \in F$, $C := uTv + e$ is a cycle and consequently every other $u - v$ edge must be contained in one of the regions of C . If the $u - v$ edge \bar{e} is contained in the bounded face of C then by Fact 2.5 there must be a vertex z from $V(T)$ which is in the bounded region of the cycle $uev\bar{e}u$. Then however z is connected either with u or with v (assume with u) by the path zTu which is contained in the bounded face of $uev\bar{e}u$. Consequently e and \bar{e} end in different sides of u . Now among all $u - v$ edges from F select at most two edges e, e' such that any other edge from F is contained in the bounded face of $ueve'u$. \square

Let H be the supergraph of T obtained as follows. For every vertex $v \in V(T)$ put a vertex w_v in each side of v and join it with v by one edge. Substitute an edge from $E(M_G) \setminus E(T)$ which ends in the side of v containing w_v with the edge that ends in w_v . Then H is a plane multigraph with $\|H\| = \|M_G\|$ which by Claim 2.7 has at most two edges between any two vertices. In addition, $|H| = |T| + \sum d_T(v) = 3|T| - 2 = 3|M_G| - 2$. Thus by Euler's formula

$$\|M_G\| = \|H\| \leq 6|H| - 12 = 18|M_G| - 24.$$

\square

2.2 Separation Procedure

Similarly as in the previous section, assume that two partitions $\mathcal{P} = (P_1, \dots, P_k)$, $\mathcal{C} = (C_1, \dots, C_l)$ of the vertex set V are given and recall that sets C_i are called small clusters, P_j are called big clusters, and T_i denotes the BSF tree rooted at d_i in $G[C_i]$. We will need a few additional concepts before we describe our first algorithm.

DEFINITION 2.8. *Let $\mathcal{P} = (P_1, \dots, P_k)$, $\mathcal{C} = (C_1, \dots, C_l)$ be two partitions of vertex set V with \mathcal{C} being a refinement of \mathcal{P} . Suppose that $C_i \subseteq P_j$. Then*

- (a) $\partial(C_i)$ denotes the set of vertices in C_i that have a neighbor in $V \setminus C_i$;
- (b) $\partial_{IN}(C_i)$ denotes the subset of $\partial(C_i)$ with vertices that have a neighbor in $P_j \setminus C_i$;
- (c) $\partial_{OUT}(C_i)$ denotes the subset of $\partial(C_i)$ with vertices that have a neighbor in $V \setminus P_j$;
- (d) $\partial^*(P_j)$ denotes the set $\{C_i | C_i \subseteq P_j \text{ and } \partial_{OUT}(C_i) \neq \emptyset\}$.

Note that $\partial_{IN}(C_i) \cap \partial_{OUT}(C_i)$ can be non-empty. We can now describe the first procedure of this section. The main objective of it is to find a "small" subset of vertices that separates the interior of each big cluster from the rest of the graph.

PROCEDURE SEPARATION

Input: Connected planar graph G and an l -dominating set $D = \{d_1, \dots, d_l\}$ in G . Partitions $\mathcal{P} = (P_1, \dots, P_k)$, $\mathcal{C} = (C_1, \dots, C_l)$ of $V(G)$ as in Definition 2.1 and (T_1, \dots, T_l) where T_i is the BFS tree in $G[C_i]$ rooted at d_i .

Output: Set $Z \subset V(G)$ and pairwise disjoint subsets (B_1, \dots, B_m) such that every $G[B_i]$ is connected.

1. Every vertex $v \in V(G) \setminus D$ sets $color[v] := white$.
2. For every P_j and every $C_i \in \partial^*(P_j)$ in parallel:

- (a) If $v \in (\partial_{IN}(C_i) \setminus \{d_i\})$ then $color[v] := blue$ and if w is on the unique path vT_id_i and $w \neq d_i$ then $color[w] := blue$.
- (b) For $v \in (\partial_{OUT}(C_i) \setminus \{d_i\})$, $color[v] := green$ when $v \in \partial_{IN}(C_i) \cap \partial_{OUT}(C_i)$ and $color[v] := red$ otherwise. Repeat the following steps in C_i until no vertex of C_i changes its color.
- * If $color[v] = green$ and w , such that $w \neq d_i$, is on the unique path vT_id_i then $color[w] := green$.
 - * If $color[v] = white$ and some $w \in N(v)$ has $color[w] = red$ then $color[v] := red$.
 - * If $color[v] = blue$ and some $w \in N(v)$ has $color[w] = red$ then $color[v] := green$.
3. Let Z be the set of vertices v with $color[v] = green$. For a cluster P_j let $Blue_j$ ($White_j$) be the set of blue (white) vertices in P_j , let $Black_j$ be the subset of $D \cap P_j$ of vertices which are not C_i 's from $\partial^*(P_j)$, and let $int(P_j) = Blue_j \cup White_j \cup Black_j$. Let $int(P_j^1), \dots, int(P_j^{l_j})$ be vertex sets of connected components of $G[int(P_j)]$.
4. Return Z and $(int(P_1^1), \dots, int(P_k^{l_k}))$.

LEMMA 2.9. *Let \tilde{G} be any plane drawing of a planar graph G . When SEPARATION finishes and v has $color[v] = green$ then v lies on a boundary path of some bunch \mathcal{B} in \tilde{G} .*

Proof. By the way of contradiction suppose that $v \in C_i \subset P_t$ has $color[v] = green$ and v does not lie on a boundary path of some bunch. We will first show that v lies on a special path. Since $color[v] = green$, for some $v' \in C_i$, vT_iv' is a path and either (1) $v' \in \partial_{IN}(C_i) \cap \partial_{OUT}(C_i)$ or (2) $color[v'] = blue$ in step 2(a) of SEPARATION. In the first case, v' has a neighbor $w \in C_j \subset P_s$ with $s \neq t$ and then $d_iT_iv'wT_jd_j$ is a special path containing v . In the second case, v' has a neighbor $w \in C_j \subset P_t$ with $j \neq i$ and $d_iT_iv'wT_jd_j$ is a special path containing v .

Now we prove that v is in a border path of some bunch. Suppose v is in a special $d_i - d_j$ path P' and let P, Q be two border paths of the bunch \mathcal{B} that contains P' . Since $v \notin V(P) \cup V(Q)$ and P, Q are border paths, v is contained in the bounded region of $P \cup Q$. Consequently all of the neighbors of v are in $Reg[P \cup Q]$ and no path of the form wT_id_i with $w \notin F[P \cup Q]$ contains v . First suppose that the second endpoint of P' , d_j , is in P_s with $s \neq t$. Since $color[v] = green$, there must be a vertex in $F[P \cup Q] \cap C_i$ which is

blue in step 2(a) and for this to be the case the set $F[P \cup Q] \cap D$ must be non-empty which contradicts the definition of a bunch. Now assume that $d_j \in C_j \subseteq P_t$. Then all vertices from $(V(P) \cup V(Q)) \cap C_i$ but d_i are colored with $blue$ in 2(a) and consequently they will never be red. Therefore for v to acquire color $green$ there must be a red vertex in $F[P \cup Q]$ and so $F[P \cup Q] \cap D$ cannot be empty contradicting the definition of a bunch. \square

LEMMA 2.10. *Let G be a connected planar graph and let D be an l -dominating set with $|D| \geq 2$. Then the set Z returned by SEPARATION has*

$$|Z| \leq 72 \cdot l|D| - 96 \cdot l.$$

Proof. Since D is l -dominating every special path has at most $2l$ vertices. In a plane drawing of G a bunch contains at most two border paths and from Lemma 2.9 every green vertex is on a border path of a bunch. Therefore, the number of green vertices is at most $4l|\mathbf{B}|$ where \mathbf{B} is the set of bunches in the drawing of G . From Fact 2.6 and equation (1) we see that

$$|Z| \leq 4l(18|D| - 24) = 72 \cdot l|D| - 96 \cdot l.$$

Finally, we establish the separation property.

LEMMA 2.11. *Every edge e which has exactly one endpoint in $int(P_j)$ has the second endpoint in $Z \cup D$.*

Proof. Let $e = \{u, v\}$ be such that v is in $int(P_j)$ and $u \notin int(P_j)$. If v is in a small cluster which is not in $\partial^*(P_j)$ then $u \in C_i$ for some $C_i \in \partial^*(P_j)$ and by step 2(a) of SEPARATION either $u = d_i$ or in 2(a) $color[u] = blue$. Since blue vertices are in $int(P_j)$ the color of u cannot be blue after the procedure finishes and so it is green. If v is in a small cluster in $\partial^*(P_j)$ then by definition $color[v] = blue$ or $color[v] = white$ after the procedure finishes. Clearly v cannot have a red neighbor as otherwise its color would change nor can $u \in V \setminus P_j$ as then v is again either green or red. Since $u \notin int(P_j)$, $color[u]$ cannot be $blue$ nor $white$ and since it cannot be red it must be green or black. Thus $color[u] = green$ or $u = d_i$.

3. PACKING PROBLEMS

In this section we will give the main packing algorithm. In addition, in Section 3.2, we will comment on a useful generalization of the packing problem that can be solved using the exactly same procedure.

3.1 Packing Algorithm

Recall that in a graph $G = (V, E)$ a subset $S \subset V$ is called an (a, b) -ruling set if it is a b -dominating set

and for every $v, v' \in S$ if $v \neq v'$ then $\text{dist}(v, v') \geq a$. A simple distributed procedure from [1] finds a $(k, O(k \log |G|))$ -ruling set in $O(k \log |G|)$ rounds. Note that an (a, b) -ruling set S can be used to obtain the ruling set partition of $V(G)$. Indeed, every vertex from $V(G) \setminus S$ joins a group with a vertex from S which is closest to it (conflicts resolved arbitrarily). The resulting partition (C_1, \dots, C_l) is such that every C_i contains exactly one vertex from S , and $G[C_i]$ has radius b . In addition to a ruling set partition we will need partitioning procedures from [3] and [5]. Moreover we will be invoking the procedure from [5] in a weighted planar graphs with weights on vertices. Let us first very briefly define the properties of a partition from [5]. Let $G = (V, E)$ be a planar graph and let $\omega : V \rightarrow \mathbb{R}^+ \cup \{0\}$ be a weight function. Recall that for a partition (P_1, \dots, P_k) of $V(G)$ we use $\partial(P_i)$ to denote the subset of P_i containing vertices which have a neighbor in $V \setminus P_i$. For a set $U \subseteq V(G)$ we use $\omega(U)$ to denote $\sum_{u \in U} \omega(u)$.

DEFINITION 3.1. *If $G = (V, E, \omega)$ is as above then a partition (P_1, \dots, P_k) of V is called an (α, β) -partition if*

- (a) $\sum_{i=1}^k \omega(\partial(P_i)) \leq \beta \omega(V(G))$ and
- (b) for each i , the weak diameter of $G[P_i]$ is at most α and $G[P_i]$ is a connected graph.

In [5] it is shown that a $(\text{polylog}(m), 1/\log^{\Theta(1)} m)$ -partition can be found in $\text{polylog}(m)$ rounds when G is weighted planar graph with $|G| \leq m$. In addition, if $\omega(v) \in \{0, 1\}$ then, as proved in [3] a $(O(1), 1/2)$ -partition can be found in $\text{polylog}(m)$ rounds and one can guarantee that the diameter of each $G[C_i]$ is at most α . We will now proceed with the algorithm that uses SEPARATION to find a useful partition of G .

PROCEDURE CLUSTERS

Input: Planar graph G and $c \geq 1$.

Output: Partition \mathcal{B} of $V(G)$ and set $Z \subset V(G)$.

1. Find a $(2c + 1, O(c \log |G|))$ -ruling set D in G and let $D^{(0)} := D, G^{(0)} := G$.
2. For $a := 1$ to $O(\log |G|)$ do
 - (a) Find the ruling set partition $(C_1^{(a)}, \dots, C_{l_a}^{(a)})$ of $G^{(a-1)}$ around $D^{(a-1)}$ and contract each of the C_i 's to a single vertex to obtain the minor $F^{(a-1)}$.
 - (b) Use the procedure from [3] to find a $(O(1), 1/2)$ -partition $\bar{\mathcal{P}}$ of $F^{(a-1)}$. Let $(P_1^{(a)}, \dots, P_{s_a}^{(a)})$ be the partition of $G^{(a-1)}$ obtained from $\bar{\mathcal{P}}$ by uncontracting vertices of $F^{(a-1)}$.

- (c) Use SEPARATION with partitions $(P_1^{(a)}, \dots, P_{s_a}^{(a)})$, $(C_1^{(a)}, \dots, C_{l_a}^{(a)})$ and with $T_i^{(a)}$ being the BFS tree rooted at $d_i \in D^{(a-1)} \cap C_i^{(a)}$. SEPARATION returns $Z^{(a)}$ and the family of sets $\{B_i^{(a)}\}$.
- (d) Let $V := V \setminus \bigcup_i B_i^{(a)}, G^{(a)} := G[V], D^{(a)} := D^{(a-1)} \cap V$.

3. Let $Z := \bigcup (Z^{(a)} \cup D^{(a)})$. Return Z and the collection of all sets $B_i^{(a)}$ for all a and i .
-

LEMMA 3.2. *After each iteration of the loop in step two of CLUSTERS the following conditions are satisfied.*

- (a) Every edge in $G^{(a)}$ with exactly one endpoint in $B_i^{(a)}$ has the second endpoint in Z .
- (b) $|Z^{(a)}| = O(c|D| \log |G|/2^{a-1})$.
- (c) Set $D^{(a)}$ is a $O(c \log |G|)$ -dominating set in $G^{(a)}$.

Proof. Part (a) follows immediately from Lemma 2.11. For part (b) note that $|D^{(a+1)}| = |\partial(\bar{\mathcal{P}})| \leq |D^{(a)}|/2$ and so $|D^{(a+1)}| \leq |D|/2^{a+1}$. From Lemma 2.10 applied with $l = O(c \log |G|)$, we have

$$|Z^{(a)}| = O(c \log |G| |D^{(a-1)}|) = O(c|D| \log |G|/2^{a-1}).$$

To establish (c) we will show that $D^{(a)}$ is $O(c \log |G|)$ -dominating in the graph induced by green and red vertices. To that end suppose that $v \in C_i \in \partial^*(P_j)$ and $\text{color}[v] \in \{\text{green}, \text{red}\}$. Then every vertex w on the path $vT_i d_i$ in BSF T from v to the root d_i has $\text{color}[w] \in \{\text{green}, \text{red}\}$ unless $w = d_i$. Indeed suppose that w is such that $\text{color}[w] \in \{\text{white}, \text{blue}\}$ and such that the length of $vT_i w$ is the smallest. If $vT_i u w T_i d_i$ is the $d - d_i$ path in T_i then $\text{color}[u] = \text{green}$ as $u \notin D^{(a-1)}$ but then in SEPARATION step 2(b) we would set $\text{color}[w]$ to be *green* or *red*. A vertex v is in $V(G^{(a)})$ if by the end of iteration $a - 1$ $v \in C_i$ where $C_i \in \partial^*(P_j)$ and $\text{color}[v]$ is *green* or *red*. Therefore if $v \in V(G^{(a)})$ then all vertices on the path $vT_i d_i$ are in $V(G^{(a)})$ and the distance in $G^{(a)}$ between v and d_i is $O(c \log |G|)$. \square

LEMMA 3.3. *Let $G = (V, E)$ be the input graph to CLUSTERS and let (B_1, \dots, B_q) be the family of sets returned in step three. Then the following conditions are satisfied.*

- (a) (B_1, \dots, B_q) is a partition of V .
- (b) Set Z is a vertex cover of the set of all cross-edges in (B_1, \dots, B_q) .
- (c) $|Z| = O(c|D| \log |G|)$ where D is the set in step one of CLUSTERS.

Proof. To prove part (a), note that in step 2(b) of CLUSTERS, a partition found in the second step is such that the number of small clusters $C_i^{(a)}$ which are in $\bigcup_j \partial^*(P_j^{(a)})$ is at most $l_a/2$. Consequently in the a th iteration we have $|D^{(a)}| \leq |D|/2^a$ and after $O(\log |G|)$ iterations the set V will be empty and every vertex of G is in one of the sets $B_i^{(a)}$. For part (b), note that every cross-edge of (B_1, \dots, B_q) is an edge with exactly one endpoint in some $B_i^{(a)}$ and so by Lemma 3.2 part (a) its second endpoint is in Z . Finally to show (c) note that there are $O(\log |G|)$ iterations and from Lemma 3.2, $|Z| = O(c|D| \log |G| \sum_{a \geq 0} 2^{-a}) = O(c|D| \log |G|)$. \square

In the next algorithm we first use Z to define weights on vertices of special minor and then invoke the clustering procedure from [5] to partition the graph. Finally we find optimal solutions locally and return the union of them.

PACKING

Input: Connected graph H , planar graph G , and a positive integer k .

Output: Packing of H in G .

1. For every vertex v in G if there is no subgraph of G which is isomorphic to H and which contains v then delete v from G .
2. Call CLUSTERS with $c = |H|$ and let $B = (B_1, \dots, B_q)$ be the partition and Z be the set returned by CLUSTERS.
3. Let G' be the weighted minor of G obtained by contracting every B_i to vertex b_i and setting $\omega(b_i) := |B_i \cap Z|$.
4. Find a $O(\text{polylog}(|G|), 1/\log^{k+1} |G|)$ -partition A' of G' using the algorithm from [5] and let $A = (A_1, \dots, A_t)$ be the partition of G obtained from A' by uncontracting b_i 's.
5. Find an optimal packing of H in each of $G[A_i]$'s and return the union.

THEOREM 3.4. *Let $\nu_H(G)$ be the value of an optimal packing of H in G and let k be a fixed constant.*

Given H and k PACKING returns a packing of H in G value of which is at least $(1 - O(|H|/\log^k |G|))\nu_H(G)$. The algorithm runs in a $\text{polylog}(|G|)$ number of rounds provided $|H| = \text{polylog}(|G|)$ and $k = O(1)$.

Proof. Let \mathcal{H} be a packing of H of size $\nu(H)$. Let \mathcal{H}_{IN} be the subset of \mathcal{H} containing H from \mathcal{H} such that H is a subgraph of $G[A_i]$ for some i . Since in the fifth step of the algorithm, an optimal packing in each of $G[A_i]$'s is found the solution returned by the algorithm has the value of at least $|\mathcal{H}_{IN}|$. Let $\mathcal{H}_{OUT} = \mathcal{H} \setminus \mathcal{H}_{IN}$. As graphs from \mathcal{H} are vertex-disjoint subgraphs of G the size of \mathcal{H}_{OUT} is at most the size of the maximum matching in $G[\partial(A)]$. Every edge from $G[\partial(A)]$ has one of its endpoints in Z as B is a refinement of A and by Lemma 3.3 part (b), Z is a vertex cover of $G[\partial(B)]$. Let Z' be the subset of Z which is a vertex cover of $G[\partial(A)]$. Then, using Lemma 3.3 part (c),

$$\begin{aligned} |Z'| &\leq \omega(G')/\log^{k+1} |G| = |Z|/\log^{k+1} |G| \\ &\leq c|D|/\log^k |G| \end{aligned}$$

where $c = |H|$. As the size of the maximum matching is at most the size of the minimum cover, we have

$$|\mathcal{H}_{OUT}| = O(c|D|/\log^k |G|).$$

In addition, $|D| \leq \nu_H(G)$ as any two distinct vertices in D are at distance of at least $2c+1$ and each vertex in the graph is contained in a copy of H . Thus the solution returned by the algorithm is of size which is at least

$$\nu_H(G) - O(\nu_H(G)c/\log^k |G|) = \left(1 - \frac{|H|}{\log^k |G|}\right) \nu_H(G).$$

\square

3.2 Packing of allowable subgraphs

Instead of packing just one graph H it is sometimes convenient to consider a packing of a family of graphs \mathcal{H} . The algorithm from the previous section can be adopted to this problem provided every subgraph from \mathcal{H} has size bounded by $\text{polylog}(|G|)$. Moreover, in some applications of packing-related problems, it is necessary to allow only certain types of subgraphs. For example, if one wants to find a maximum number of paths connecting two sets A and B then only paths that start in A and end in B are allowable subgraphs. Again, the algorithm from the previous section can be adopted to this variant with almost no changes provided allowable graphs have sizes bounded by $\text{polylog}(|G|)$. We will finish this short section by giving two more examples of allowable subgraphs packing. In the first problem, which is related to linkability (see [8]), we are give k pairs of vertices (s_i, t_i) and want to find a maximum number of $s_i - t_i$ paths of a constant length in the graph. In the second problem

every node v in a graph desires $q(v)$ neighbors assigned to it to perform concurrent computations and we want to fulfill the maximum number of requests. In this example the allowable subgraphs are stars.

4. REFERENCES

- [1] B. Awerbuch, A. V. Goldberg, M. Luby, and S. A. Plotkin. Network decomposition and locality in distributed computation. *Proc. 30th IEEE Symp. on Foundations of Computer Science*, pages 364–369, 1989.
- [2] R. Cole and U. Vishkin. Deterministic coin tossing with applications to optimal parallel list ranking. *Inf. Control*, 70(1):32–53, 1986.
- [3] A. Czygrinow and M. Hanckowiak. Distributed almost exact approximations for minor-closed families. In Y. Azar and T. Erlebach, editors, *ESA*, volume 4168 of *Lecture Notes in Computer Science*, pages 244–255. Springer, 2006.
- [4] A. Czygrinow and M. Hanckowiak. Distributed approximation algorithms in unit-disk graphs. In S. Dolev, editor, *DISC*, volume 4167 of *Lecture Notes in Computer Science*, pages 385–398. Springer, 2006.
- [5] A. Czygrinow and M. Hanckowiak. Distributed approximation algorithms for weighted problems in minor-closed families. In G. Lin, editor, *COCOON*, volume 4598 of *Lecture Notes in Computer Science*, pages 515–525. Springer, 2007.
- [6] A. Czygrinow and M. Hanckowiak. Distributed approximations for packing in unit-disk graphs. In A. Pelc, editor, *DISC*, volume 4731 of *Lecture Notes in Computer Science*, pages 152–164. Springer, 2007.
- [7] A. Czygrinow, M. Hanckowiak, and E. Szymanska. Distributed approximation algorithms for planar graphs. In T. Calamoneri, I. Finocchi, and G. F. Italiano, editors, *CIAC*, volume 3998 of *Lecture Notes in Computer Science*, pages 296–307. Springer, 2006.
- [8] R. Diestel. *Graph Theory, third (electronic) edition*, volume 173 of *New York Graduate Texts in Mathematics*. Springer, 2005.
- [9] M. Elkin. Distributed approximation: a survey. *SIGACT News*, 35(4):40–57, 2004.
- [10] M. Hańćkowiak, M. Karoński, and A. Panconesi. A faster distributed algorithm for computing maximal matchings deterministically. In *PODC '99: Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing*, pages 219–228, New York, NY, USA, 1999. ACM.
- [11] F. Kuhn, T. Moscibroda, T. Nieberg, and R. Wattenhofer. Fast deterministic distributed maximal independent set computation on growth-bounded graphs. In P. Fraigniaud, editor, *Distributed algorithms*, volume 3724 of *Lecture Notes In Computer Science*, pages 273–287, September 2005.
- [12] F. Kuhn, T. Nieberg, T. Moscibroda, and R. Wattenhofer. Local approximation schemes for ad hoc and sensor networks. In *DIALM-POMC '05: Proceedings of the 2005 joint workshop on Foundations of mobile computing*, pages 97–103, New York, NY, USA, 2005. ACM.
- [13] N. Linial. Locality in distributed graph algorithms. *SIAM J. Comput.*, 21(1):193–201, 1992.
- [14] D. Peleg. *Distributed computing: a locality-sensitive approach*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.