GRAD AND CLASSES WITH BOUNDED EXPANSION II. ALGORITHMIC ASPECTS

JAROSLAV NEŠETŘIL AND PATRICE OSSONA DE MENDEZ

Abstract. Classes of graphs with *bounded expansion* are a generalization of both proper minor closed classes and degree bounded classes. Such classes are based on a new invariant, the *greatest reduced average density (grad) of G with rank r*, $\nabla_r(G)$. These classes are also characterized by the existence of several partition results such as the existence of low tree-width and low tree-depth colorings [19][18]. These results lead to several new linear time algorithms, such as an algorithm for counting all the isomorphs of a fixed graph in an input graph or an algorithm for checking whether there exists a subset of vertices of *a priori* bounded size such that the subgraph induced by this subset satisfies some arbirtrary but fixed first order sentence. We also show that for fixed p, computing the distances between two vertices up to distance p may be performed in constant time per query after a linear time preprocessing. We also show, extending several earlier results, that a class of graphs has sublinear separators if it has sub-exponential expansion. This result result is best possible in general.

1. Introduction

The concept of tree-width [15],[24],[27] is central to the analysis of graphs with forbidden minors done by Robertson and Seymour and gained much algorithmic attention thanks to the general complexity result of Courcelle about monadic second-order logic graph properties decidability for graphs with bounded tree-width [6],[7]. It appeared that many NP-complete problems may be solved in polynomial time when restricted to a class with bounded tree-width. This restriction of tree-width is quite a strong one, as it does not include the class of planar graphs, for instance.

Another way is to consider partitions of graphs into parts such that any p of them induce a graph with low tree-width. DeVos et al. [8] proved that for any proper minor closed class of graphs $\mathcal{C}$ — that is: any minor closed class of graphs excluding at least one minor — and any integer p, there exists a constant $N(\mathcal{C}, p)$ so that any graph $G \in \mathcal{C}$ has a vertex-partition into at most $N(\mathcal{C}, p)$ parts such that any $i \leq p$ parts induce a graph of tree-width at most $(i - 1)$.

Supported by grant 1M0021620808 of the Czech Ministry of Education.

It is then natural to ask whether the parts could be choosen even "smaller" or "simple". This issue has been studied in [20] where the authors introduce the *tree-depth* $\mathrm{td}(G)$ of a graph G as the minimum height of a rooted forest including the graph in its closure. This minor monotone invariant is related to tree-width by $\mathrm{tw}(G) + 1 \leq \mathrm{td}(G) \leq \mathrm{tw}(G) \log n$, where n is the order of G. The class of graphs with bounded tree-depth appears to be particularly small, as it includes only a bounded number of rigid graphs (that is: graphs having no non-trivial automorphisms) and as it excludes long paths (to compare with classes with bounded tree-width which exclude big grids). The main result of [20] is that for any proper minor closed class of graphs $\mathcal{C}$ and any integer p, there exists an integer $N'(\mathcal{C}, p)$ such that any graph $G \in \mathcal{C}$ has a vertex-partition into at most $N'(\mathcal{C}, p)$ parts such that any $i \leq p$ parts induce a graph of tree-depth at most i. It is also proved in [20] that the tree-depth is the greatest graph invariant for which such a statement holds.

Our first proof [20] of this decomposition result relied in the result of DeVos et al. and thus indirectly to the Structural Theorem of Robertson and Seymour [25]. However since then, we generalized these results [19][18] to classes with bounded expansion (which may be seen as a generalization of both proper minor closed classes and degree bounded classes). Our prrof is both more general and conceptually easier. Even better: it leads to a linear time algorithm that we shall describe here. Our main goal will be then to show that this algorithm has a wide range of algorithmic applications.

Before we shall consider algorithmic consequences, we shall introduce bounded expansion and related concepts in Section 2.

In Section 4 we describe the augmentation process which is the basis of the partition theorem and propose a linear time algorithm for it.

2. The grad of a graph and classes with bounded expansion

The *distance* $d(x, y)$ between two vertices x and y of a graph is the minimum length of a path linking x and y, or ∞ if x and y do not belong to the same connected component. The *radius* $\rho(G)$ of a connected graph G is: $\rho(G) = \min_{r \in V(G)} \max_{x \in V(G)} \mathrm{d}(r, x)$

Definition 2.1. Let G be a graph. A *ball* of G is a subset of vertices inducing a connected subgraph. The set of all the families of balls of G is noted $\mathfrak{B}(G)$. The set of all the families of balls of G including no two intersecting balls is noted $\mathfrak{B}_1(G)$.

Let $\mathcal{P} = \{V_1, \ldots, V_p\}$ be a family of balls of G.

- The *radius* $\rho(\mathcal{P})$ of $\mathcal{P}$ is $\rho(\mathcal{P}) = \max_{X \in \mathcal{P}} \rho(G[X])$

- The *quotient* $G/\mathcal{P}$ of G by $\mathcal{P}$ is a graph with vertex set $\{1,\dots,p\}$ and edge set $E(G/\mathcal{P}) = \{\{i,j\} : (V_i \times V_j) \cap E(G) \neq \emptyset \text{ or } V_i \cap V_j \neq \emptyset\}$.

Definition 2.2. The *greatest reduced average density (grad) of* G *with rank* r is

$$\nabla_r(G) = \max_{\substack{\mathcal{P}\in\mathfrak{B}_1(G)\\ \rho(\mathcal{P})\leq r}} \frac{|E(G/\mathcal{P})|}{|\mathcal{P}|}$$

The first grad, ∇_0 , is closely related to degeneracy (G is k-degenerated iff $k \geq \lfloor 2\nabla_0(G)\rfloor$). The grads of a graph form an increasing sequence which becomes constant starting from some index (smaller than the order of the graph).

Definition 2.3. A class of graphs $\mathcal{C}$ has *bounded expansion* if there exists a function $f : \mathbb{N} \to \mathbb{R}$ such that for every graph $G \in \mathcal{C}$ and every r holds

$$\nabla_r(G) \leq f(r) \tag{1}$$

Here are some examples of class with bounded expansion:

Example 1. Any proper minor closed class of graphs has expansion bounded by a constant function. Conversely, any class of graphs with expansion bounded by a constant is included in some proper minor closed class of graphs.

Proof. If $\mathcal{C}$ is a proper minor closed class of graph, the graphs in $\mathcal{C}$ are k-degenerated for some integer k hence $\nabla_r(G) \leq k+1$ for any $G \in \mathcal{C}$.

Conversely, assume $\mathcal{C}$ is a class of graph with expansion bounded by a constant C. Let $\mathcal{C}'$ be the class defined by $\mathcal{C}' = \{G : \forall r \geq 0, \nabla_r(G) \leq C\}$. This class obviously includes $\mathcal{C}$. Let $G \in \mathcal{C}'$ and let H be a minor of G. Then for any $r \geq 0$, $\nabla_r(H) \leq \nabla_{|V(G)|}(G) \leq C$ thus $H \in \mathcal{C}'$. Hence $\mathcal{C}'$ is a proper minor closed class as it does not include K_{2C+2} (as $\nabla_0(K_{2C+2}) = C+1$). □

Example 2. Let Δ be an integer. Then the class of graphs with maximum degree at most Δ has expansion bounded by the exponential function $f(r) = \Delta^{r+1}$.

Example 3. In [17] is introduced a class of graphs which occurs naturally in finite-element and finite-difference problems. These graphs correspond to graphs embedded in d-dimensional space in a certain manner. It is proved in [26] that these graphs excludes K_h as a depth L minor if $h = \Omega(L^d)$. Hence they form (for each d) a class with polynomialy bounded expansion.

The next example show that the bounded function can be any arbitrary increasing function:

Example 4. Let f be any increasing function from $\mathbb{N}$ to $\mathbb{N} \setminus \{0, 1, 2\}$. Then there exists a class $\mathcal{C}$ such that $\mathcal{C}$ has expansion bounded by f but by no smaller integral function.

Proof. Consider the class $\mathcal{C}$ whose elements are K_4 and the graphs G_n obtained by subdividing $2n$ times the complete graph $K_{2f(n)+1}$ (for $n \geq 1$). As $2 \leq \nabla_r(G_n) < 3$ for $r < n$ and as $\nabla_r(G_n) = f(n)$ for $r \geq n$, we conclude. □

Example 5. If $\mathcal{C}$ is a class with bounded expansion and if c is any fixed integer then the class $\mathcal{C}' \bullet K_c$ whose elements are the lexicographic products $G \bullet K_c, G \in \mathcal{C}$ still has bounded expansion [18].

It should be noted that such a statement is false for proper minor closed classes in a strong sense: for any $n \in \mathbb{N}$, K_n is a minor of $\mathrm{Grid}(2n, 2n) \bullet K_2$.

2.1. **Few properties of tree-depth.** A *rooted forest* is a disjoint union of rooted trees. The *height* of a vertex x in a rooted forest F is the number of vertices of a path from the root (of the tree to which x belongs to) to x and is noted $\mathrm{height}(x, F)$. The *height* of F is the maximum height of the vertices of F. Let x, y be vertices of F. The vertex x is an *ancestor* of y in F if x belongs to the path linking y and the root of the tree of F to which y belongs to. The *closure* $\mathrm{clos}(F)$ of a rooted forest F is the graph with vertex set $V(F)$ and edge set $\{\{x, y\} : x \text{ is an ancestor of } y \text{ in } F, x \neq y\}$. A rooted forest F defines a partial order on its set of vertices: $x \leq_F y$ if x is an ancestor of y in F. The comparability graph of this partial order is obviously $\mathrm{clos}(F)$.

Definition 2.4. The *tree-depth* $\mathrm{td}(G)$ of a graph G is the minimum height of a rooted forest F such that $G \subseteq \mathrm{clos}(F)$.

Lemma 2.1. *Let G be a connected graph with maximum degree Δ and tree-depth $t \geq 1$. Then G has order $n \leq 1 + \Delta + \cdots + \Delta^{t-1}$.*

Proof. We proceed by induction over t. If $t = 1$, $G = K_1$ and $\mathrm{td}\, K_1 = 1$. Assume the inequality has been proved for graphs with tree-depth at most $(t-1)$ with $t \geq 2$ and let G be a connected graph with tree-depth t. As G is connected it contains a vertex r such that $\mathrm{td}(G - r) = \mathrm{td}(G) - 1 = t - 1$. Let $G_1, \ldots, G_k$ be the connected components of $G - r$. All of these have tree-depth at most $t - 1$. By induction, they have order at most $1 + \Delta + \cdots + \Delta^{t-2}$. As $k \leq \Delta$, we conclude. □

Lemma 2.2. *For $n \geq 1, \mathrm{td}(P_n) = \lceil \log_2(k+1) \rceil$*

Proof. According to lemma 2.1 a path of tree-depth t has order at most $1 + 2 + \cdots + 2^{t-1} = 2^t - 1$. It follows that the tree-depth of P_k is at least $\log_2(k+1)$.

Moreover let $x_1, \ldots, x_{2^t-1}$ be the vertices of a path of order $2^t - 1$ in the order in which they appear on the path. Let $w(i)$ be the base 2 word of length t corresponding to the number i (for instance, if $t = 3$, $w(1) = 001, w(2) = 010, \dots, w(7) = 111$). Let $c(i)$ be the rank of the rightmost 1 of $w(i)$ (that is, for $t = 3$, $c(1) = c(3) = c(5) = c(7) = 3$, $c(2) = c(6) = 2$ and $c(4) = 1$). Then c is a centered coloring of P_{2^t-1} with t colors thus $\mathrm{td}(P_{2^t-1}) \le t$. Finally we note that $\mathrm{td}(P_n)$ increases with n. □

Lemma 2.3. *Let G be a graph and let P_k be the longest path in G. Then $\lceil \log_2(k+1) \rceil \le \mathrm{td}(G) \le \binom{k+2}{2} - 1$.*

Proof. As the tree-depth is minor monotone, any graph including a path P_k as a subgraph as tree-depth at least $\mathrm{td}(P_k) = \lceil \log_2(k+1) \rceil$ (according to Lemma 2.2).

Conversely, let us prove by induction over $k \ge 1$ that a graph which includes no path P_k has tree depth at most $\binom{k+1}{2}$. Obviously the statement holds for $k = 1$ (graphs without edges has tree-depth 1). Assume the statement has been proved up to $(k-1)$ for some $k \ge 2$. Let G be a graph with no path P_k. Without loss of generality we may assume that G includes a P_{k-1} and that G is connected (as the tree-depth of a non-connected graph is the maximum of the tree-depths of its connected components). Let P be such a path of G. Assume $G - V(P)$ includes some path P' isomorphic to P_{k-1}. According to the connectivity of G, there exists some minimum length path P'' linking a vertex of P to a vertex of P' (and this path has length at least 1). Then $P \cup P' \cup P''$ includes a P_k, a contradiction. Thus $G - V(P)$ includes no P_{k-1}. By induction, $\mathrm{td}(G - V(P)) \le \binom{k}{2} - 2$ hence $\mathrm{td}(G) \le \mathrm{td}(G - V(P)) + |V(P)| \le \binom{k}{2} + k = \binom{k+1}{2}$. If follows that if P_k is the longest path in G, $\mathrm{td}(G)$ is at most $\binom{k+2}{2} - 1$. □

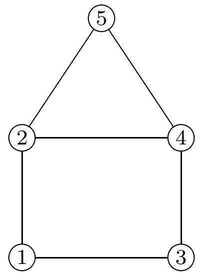
3. Basics

We shall first mention some basic linear time algorithms, as well as the basic data structures used for input and output of our algorithms. Concerning the data structure used for the computations, any standard one will do, but we will have in mind the simple data structure of `PIGALE` library [12][13].

3.1. **Graph representation of input and output graphs.**

It will be convenient for our algorithms to represent graphs as follows:

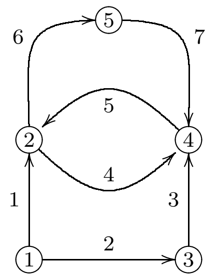
- The vertices of the input graph will be assumed to be numbered from 1 to n and its edges will be given as a list L of pairs (i, j). Although this representation allows multigraphs and loops, we will consider simple input graphs only. Notice that this representation may be computed in linear time from any standard one.



$$n = 5$$

$$L = ((1, 2), (1, 3), (3, 4), (4, 2), (5, 2), (5, 4))$$

- A computed directed graph $\vec{G}$ will be represented as an array D of lists indexed by integers $1, \ldots, n$. In the list $D[i]$ will be gathered all the couples (j, e) such that (j, i) is an arc of $\vec{G}$ with index $e \in \{1, \ldots, m\}$ (where m is the size of $\vec{G}$). This representation may be easily transformed into any standard one in linear time.



$$n = 5, m = 7$$

$$\begin{aligned} D[1] &= () \\ D[2] &= ((1, 1), (4, 5)) \\ D[3] &= ((1, 2)) \\ D[4] &= ((2, 4), (3, 3), (5, 7)) \\ D[5] &= ((2, 6)) \end{aligned}$$

Notice that the used representation of a directed graph $\vec{G}$ allows to answer the question "is there an arc from vertex i to vertex j" in time $O(\Delta^{-}(\vec{G}))$. Notice that this simple observation has by itself many algorithmic consequences [4].

3.2. **Low indegree orientation.** The aim of the following algorithm is to compute a low-indegree orientation of the graph with vertex set $\{1, \dots, n\}$ and list of edges L.

Lemma 3.1. *Let G be a graph of order n and size m. There is an $O(n+m)$-time algorithm which computes an acyclic orientation of G with maximum indegree $\lfloor 2\nabla_0(G)\rfloor$.*

Proof. First we compute a representation of the graph in any suitable data structure like `PIGALE`'s data structure [12]. All of this may be easily done in time $O(m)$. Then we do the following:

Ensure: D represents an orientation $\vec{G}$ of G such that $\Delta^-(\vec{G}) \leq \lfloor 2\nabla_0(G)\rfloor$.
  Let $D = ()$, let $m = 0$.
  Let $T[\,]$ be an array of lists.
  Let $d[\,]$ be an array.
  Initialize $d[v]$ with degree of v, δ with the minimum degree of the graph and Δ with the maximum degree of the graph.
  Using a bucket sort, initialize $T[d]$ as the list of the vertices with degree d.
  while $\delta < \Delta$ or $T[\delta]$ is not empty **do**
    pop v out of $T[\delta]$.
    let $d[v] \leftarrow 0$
    for all w neighbour of v **do**
      if $d[w] > 0$ **then**
        if $d[w] > \delta$ **then**
          extract w from $T[d[w]]$
          insert w in $T[d[w]-1]$
          if $d[w] = \Delta$ and $T[d[w]]$ is empty **then**
            let $\Delta \leftarrow \Delta - 1$
          end if
        end if
        let $d[w] \leftarrow d[w] - 1$
        $m \leftarrow m + 1$; append (w, m) to D.
      end if
    end for
    while $T[\delta]$ is empty and $\delta < \Delta$ **do**
      let $\delta \leftarrow \delta + 1$
    end while
  end while

In this algorithm, if δ is increased the the actual induced subgraph of G has minimum degree greater than δ. It follows that the maximum value of δ reached by the algorithm is less or equal to the maximum average degree of G, that is: $\delta \leq 2\nabla_0(G)$. It follows that this algorithm computes an acyclic orientation of G with maximum indegree $\lfloor 2\nabla_0(G)\rfloor$ in time $O(m)$. □

3.3. **Digraph simplification.** Digraph simplification may be achieved in $O(m)$ time using bucked sort [12]. We recall the algorithm here for completeness. It will be convenient for our further use to associate to each arc an integer weight and to maintain the simplification the minimum weight for parallel arcs linking two vertices.

Require: D represents a directed graph orientation $\vec{G}$ and weight[ ] is a weight array.
Ensure: D is simplified; among parallel arcs only one with minimum weight is kept.

```
D′[ ] is an array of list, Last[ ] is an array initialized with 0 and
Last_weight[ ] is an array.
for all v ∈ {1,...,n} do
  while D[v] ≠ () do
    Pop (x,e) out of D[v].
    if Last[x] = e then
      if weight[e] < Last_weight[x] then
        Pop last element of D′[x]
        Last[x] = e; Last_weight[x] = weight[e]
        Append (v,e) to D′[x]
      end if
    else
      Last[x] = e; Last_weight[x] = weight[e]
      Append (v,e) to D′[x]
    end if
  end while
end for
for all x ∈ {1,...,n} do
  while D′[x] ≠ () do
    Pop (v,e) out of D′[x]
    Append (x,e) to D[v]
  end while
end for
```

4. Transitive fraternal augmentations of graphs in linear time

4.1. **Theory.** In the following, a directed graph $\vec{G}$ may not have a loop and for any two of its vertices x and y, $\vec{G}$ includes at most one arc from x to y and at most one arc from y to x.

Definition 4.1. Let $\vec{G}$ be a directed graph. A 1-*transitive fraternal augmentation* of $\vec{G}$ is a directed graph $\vec{H}$ with the same vertex set, including all the arcs of $\vec{G}$ and such that, for any vertices x, y, z,

- if (x, z) and (z, y) are arcs of $\vec{G}$ then (x, y) is an arc of $\vec{H}$ (*transitivity*),
- if (x, z) and (y, z) are arcs of $\vec{G}$ then (x, y) or (y, x) is an arc of $\vec{H}$ (*fraternity*).

A *transitive fraternal augmentation* of a directed graph $\vec{G}$ is a sequence $\vec{G} = \vec{G}_1 \subseteq \vec{G}_2 \subseteq \cdots \subseteq \vec{G}_i \subseteq \vec{G}_{i+1} \subseteq \cdots$, such that $\vec{G}_{i+1}$ is a 1-transitive fraternal augmentation of $\vec{G}_i$ for any $i \geq 1$.

The key result of [18] claims the existence of density bounded transitive fraternal augmentations:

Lemma 4.1 (Special case of Lemma 6.1 of [18])**.** *There exists polynomials P_i $(i \geq 0)$ such that for any directed graph $\vec{G}$ and any 1-transitive fraternal augmentation $\vec{H}$ of $\vec{G}$ we have*

$$\nabla_r(H) \leq P_{2r+1}(\Delta^-(\vec{G}) + 1, \nabla_{2r+1}(G)), \tag{2}$$

where G and H stand for the simple undirected graphs underlying $\vec{G}$ and $\vec{H}$.

Although quite technical, the next result is a simple direct consequence of Lemma 4.1:

Corollary 4.2. *Let $\mathcal{C}$ be a class with expansion bounded by a function f and let $F : \mathbb{N}^2 \to \mathbb{N}$.*

Define $A(r, i)$ and $B(i)$ recursively as follows (for $i \geq 1$ and $r \geq 0$):

$$\begin{aligned} A(r, 1) &= f(r) \\ B(1) &= 2f(0) \\ A(r, i+1) &= P_{2r+1}(B(i) + 1, A(2r+1, i)) \\ B(i+1) &= F(B(i), A(0, i+1)) \end{aligned}$$

Assume $G \in \mathcal{C}$ and $\vec{G} = \vec{G}_1 \subseteq \vec{G}_2 \subseteq \cdots \subseteq \vec{G}_i \subseteq \vec{G}_{i+1} \subseteq \cdots$ is a transitive fraternal augmentation of G such that

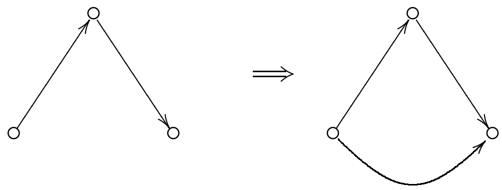
$$\Delta^-(\vec{G}_{i+1}) \leq F(\Delta^-(\vec{G}_i), \nabla_0(G_{i+1})) \text{ (for } i \geq 1)$$

and such that $\Delta^-(\vec{G}_1) \leq 2f(0)$. Then:

$$\begin{aligned} \nabla_r(G_i) &\leq A(r, i) \\ \Delta^-(\vec{G}_i) &\leq B(i) \end{aligned}$$

We now present a linear time implementation of this procedure, where it will be checked that $\Delta^-(\vec{G}_{i+1}) \leq \Delta^-(\vec{G}_i)^2 + 2\nabla_0(G_i)$, that is: $F(x, y) = x^2 + 2y$.

4.2. **The algorithm for one step augmentation.** In the augmentation process, we add two kind of arcs: transitivity arcs and fraternity arcs. Let us start with transitivity ones:

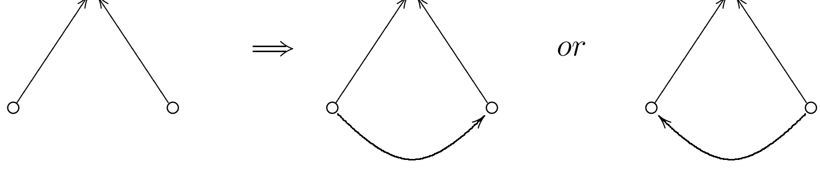


Require: D represents the directed graph to be augmented.
Ensure: D' represents the array of the added arcs.

```
Initialize D'.
Initialize an array mark[ ] with false.
for all v ∈ {1, ..., n} do
   for all (u, e) ∈ D[v] do
      for all (x, f) ∈ D[u] do
         if mark[x] = false then
            m ← m + 1; append (x, m) to D'[v].
            mark[x] ← true.
         end if
      end for
   end for
   for all (u, e) ∈ D[v] do
      for all (x, f) ∈ D[u] do
         mark[x] ← false
      end for
   end for
end for
```

This algorithm runs in $O(\Delta^-(\vec{G})^2 n)$ time, where $\Delta^-(\vec{G})$ is the maximum indegree of the graph to be augmented. It computes the list array D' of the transitivity arcs which are missing in $\vec{G}$, each missing arc appearing exactly once in the list.

Now, we shall consider the fraternity edges.



Require: D represents the directed graph to be augmented.
Ensure: L represents the list of edges to be added.

$L = ()$.
Sub[ ] is an array of list, and Last[ ] is an array initialized with 0.
for all $v \in \{1, \dots, n\}$ **do**
  for all $(x, e) \in D[v]$ **do**
    for all $(y, f) \in D[v]$ **do**
      if $x < y$ **then**
        append y to Sub$[x]$.
      end if
    end for
  end for
end for
for all $x \in \{1, \dots, n\}$ **do**
  for all $y \in$ Sub$[x]$ **do**
    if $x \neq$ Last$[y]$ **then**
      Last$[y] = x$; append (x, y) to L.
    end if
  end for
end for

This algorithm runs in $O(\Delta^-(\vec{G})^2 n)$-time and computes the list of the fraternity edges, each edge appearing exactly once.

We then use the $O(m)$-time algorithm to compute a low indegree orientation of the edges in L and we merge the obtained directed graph with D and D' (the simplification of the graphs goes the same way as in the computation of the fraternity edges, hence is done in $O(m + n)$ time, that is: $O(\Delta^-(\vec{G})^2 n)$-time).

Theorem 4.3. *For any class $\mathcal{C}$ with bounded expansion, there exists an algorithm which computes, given an input graph $G \in \mathcal{C}$, a transitive fraternal augmentation $\vec{G} = \vec{G}_1 \subseteq \vec{G}_2 \subseteq \cdots \subseteq \vec{G}_c$ of G in time $O(cn)$.*

5. Distances

The following result is a weighted extention of the basic observation that bounded orientations allows $O(1)$-time checking of adjacency [4].

Theorem 5.1. *For any class $\mathcal{C}$ with bounded expansion and for any integer k, there exists a linear time preprocessing algorithm so that for any preprocessed $G \in \mathcal{C}$ and any pair $\{x, y\}$ of vertices of G the value* $\min(k, \mathrm{dist}(x, y))$ *may be computed in $O(1)$-time.*

Proof. The proof goes by a variation of our augmentation algorithm so that each arc e gets a weight $w(e)$ and each added arc gets weight $\min(w(e_1) + w(e_2))$ over all the pairs (e_1, e_2) of arcs which may imply the addition of e and simplification should keep the minimum weighted arc.

Then, after k augmentation steps, two vertices at distance at most k have distance at most 2 in the augmented graph. The value $\min(k, \mathrm{dist}(x,y))$ then equals

$$\min\left(k, w((x,y)), w((y,x)), \min_{(z,x),(z,y)\in\vec{G}}\big(w(z,x)+w(z,y)\big)\right). \qquad \square$$

6. p-centered colorings and tree-decomposition

6.1. Theory.

Definition 6.1. A *tree-decomposition* of a graph G consists in a pair (T,λ) formed by a tree T and a function λ mapping vertices of T to subsets of $V(G)$ so that for all $v \in V(G), \{x \in V(T) : v \in \lambda(x)\}$ induces a subtree of T, and such that for any edge $\{v,w\}$ of G there exists $x \in V(T)$ such that $\{v,w\} \subseteq \lambda(x)$.

The *width* of a tree decomposition (T,λ) is $\max_{v\in V(G)}|\lambda(v)|-1$. The *tree-width* of G is the minimum width of any tree-decomposition of G.

From a rooted tree Y of height at most p such that $G \subseteq \mathrm{clos}(Y)$ it is straightforward to construct a tree-decomposition (T,λ) of G having width at most $(p-1)$: Set $T = Y$ and define $\lambda(x) = \{v \leq_Y x\}$. Then for any v, $\{x \in V(T) : v \in \lambda(x)\} = \{x \geq_Y v\}$ induces the subtree of Y rooted at v (hence a subtree of T). Moreover, as $G \subseteq \mathrm{clos}(Y)$, any edge $\{x,y\}$ with $x <_Y y$ is a subset of $\lambda(y)$. Hence (T,λ) is a tree-decomposition of G. As $\max_{v\in V(G)}|\lambda(v)| = \mathrm{height}(Y) \leq p$, this tree-decomposition has width at most $(p-1)$. Last, this tree-decomposition may be obviously constructed in linear time.

Definition 6.2. A *centered coloring* of a graph G is a coloring of the vertices such that in any connected subgraph some color appears exactly once.

For an integer p, a *p-centered coloring* of G is a coloring of the vertices such that in any connected subgraph either some color appears exactly once, or at least p different colors appear.

6.2. The algorithm.

Require: c is a centered-coloring of the graph G using colors $1,\dots,p$.
Ensure: $\mathcal{F}$ is a rooted forest such that $G \subset \mathrm{clos}(\mathcal{F})$.

Set $\mathcal{F} = \emptyset$.
Let Big[ ] be an array of size p.
for all Connected component G_i of G **do**
  Initialize Big[ ] to **false**.
  Set root_color $\leftarrow 0$.
  for all $v \in V(G_i)$ **do**
    if Big$[c[v]]$ = **false then**
      if $c[v]$ = root_color **then**

```
          root_color ← 0, Big[c[v]] ← true.
        else
          root ← v; root_color ← c[v].
        end if
      end if
    end for
    Recurse on G-root thus getting some rooted forest F′={Y′_1, . . . ,Y′_j}.
    Add to F the tree with root root and subtrees Y_1, . . . , Y_j, where
    the sons of root are the roots of Y_1, . . . , Y_j.
  end for
```

This algorithms clearly runs in $O(pm)$ time. If G is connected, it returns a rooted tree Y of height at most p such that $G \subseteq \mathrm{clos}(Y)$.

7. Application to subgraph isomorphism problem

For general subgraph isomorphism problem of deciding wether a graph G contains a subgraph isomorphic to a graph H of order l, the better known general bound is $O(n^{\alpha l/3})$ where α is the exponent of square matrix fast multiplication algorithm [21] (hence $O(n^{0.792\ l})$ using the fast matrix algorithm of [5]). The particular case of subgraph isomorphism in planar graphs have been studied by Plehn and Voigt [22], Alon [2] with superlinear bounds and then by Eppstein [9][10] who gave the first linear time algorithm for fixed pattern H and G planar and then extended his result to graphs with bounded genus [11]. We generalize this to classes with bounded expansion.

We shall now make use of the following result for graphs with bounded tree-width:

Lemma 7.1 (Eppstein, Lemma 2 of [10])**.** *Assume we are given graph G with n vertices along with a tree-decomposition T of G with width w. Let S be a subset of vertices of G, and let H be a fixed graph with at most w vertices. Then in time $2^{O(w \log w)}n$ we can count all isomorphs of H in G that include some vertex in S. We can list all such isomorphs in time $2^{O(w \log w)}n + O(kw)$, where k denotes the number of isomorphs and the term kw represents the total output size.*

We shall prove here the following extention of the results of [10][11]:

Theorem 7.2. *Let $\mathcal{C}$ be a class with bounded expansion and let H be a fixed graph. Then there exists a linear time algorithm which computes, from a pair (G, S) formed by a graph $G \in \mathcal{C}$ and a subset S of vertices of G, the number of isomorphs of H in G that include some vertex in S. There also exists an algorithm running in time $O(n) + O(k)$ listing all such isomorphism where k denotes the number of isomorphs (thus represents the output size).*

Proof. This is a direct consequence of Theorem 4.3 and Lemma 7.1. □

8. Local decidability problems

Monadic second-order logic (MSOL) is an extention of first-order logic (FOL) that includes vertex and edge sets and belonging to these sets. The following theorem of Courcelle has been applied to solve many optimization problems.

Theorem 8.1 (Courcelle [6][7]). *Let $\mathcal{K}$ be class of finite graphs $G = \langle V, E, R\rangle$ represented as τ_2-structures, that is: by two sorts of elements (vertices V and edges E) and an incidence relation R, and ϕ be a MSOL(τ_2) sentence. If $\mathcal{K}$ has bounded tree width and $G \in \mathcal{K}$, then checking wether $G \models \phi$ can be done in linear time.*

Combining Theorem 8.1 with Theorem 4.3, we get:

Theorem 8.2. *Let $\mathcal{C}$ be a class with bounded expansion and let p be a fixed integer. Let ϕ be a FOL(τ_2) sentence. Then there exists a linear time algorithms to check $\exists X : (|X| \leq p) \wedge (G[X] \models \phi)$.*

Thus for instance:

Theorem 8.3. *Let $\mathcal{K}$ be a class with bounded expansion and let H be a fixed graph. Then, for each of the next properties there exists a linear time algorithm to decide whether a graph $G \in \mathcal{K}$ satisfies them:*

- *H has a homomorphism to G,*
- *H is a subgraph of G,*
- *H is an induced subgraph of G.*

Although there is an (easy) polynomial algorithm to decide whether $td(G) \leq k$ for any fixed k, if P≠NP then no polynomial time approximation algorithm for the tree-depth can guarantee an error bounded by n^ϵ, where ϵ is a constant with $0 < \epsilon < 1$ and n is the order of the graph [3]. We shall now prove that the decision problem $td(G) \leq k$ for any fixed k may actually be decided in linear time:

Lemma 8.4. *Any Depth-First Search (DFS) tree Y of connected graph G satisfies:*

- *$G \subseteq \mathrm{clos}(Y)$,*
- *$\mathrm{td}(G) \leq \mathrm{height}(Y) \leq 2^{\mathrm{td}(G)} - 1$.*

Proof. According to the basic properties of the DFS, a vertex v of G may not be adjacent in G to a vertex which is not comparable to v with respect to the tree order induced by the DFS tree Y thus $G \subseteq \mathrm{clos}(Y)$ and $\mathrm{td}(G) \leq \mathrm{height}(Y)$. Moreover, G includes $P_{\mathrm{height}(Y)}$ as a subgraph (take any maximal tree chain) thus $\mathrm{height}(Y) \leq 2^{\mathrm{td}(P_{\mathrm{height}(Y)})} - 1$, according to Lemma 2.2. Hence $\mathrm{height}(Y) \leq 2^{\mathrm{td}(G)} - 1$ as $\mathrm{td}(P_{\mathrm{height}(Y)}) \leq \mathrm{td}(G)$. □

Theorem 8.5. *For any fixed k, there exists a linear time algorithm which decides wether an input graph G has tree-depth at most k or not.*

Proof. Without loss of generality we may assume G is connected (for otherwise we process all the connected components one by one). Any DFS tree Y of G may be computed in $O(m)$ time, where m is the size of G. If height$(Y) \geq 2^k$, the answer is "No" according to Lemma 8.4. Otherwise, consider the following sentence Φ:

$$\begin{aligned}\exists V_1 \exists V_2 \dots \exists V_k \colon & (\forall x \in V_1\ \forall y \in V_2, x \neq y) \wedge \dots \\ & \wedge (\forall x (\exists y \in V_1, x = y) \vee \dots) \\ & \wedge (\forall A (\exists B (\forall x \in A\ (x \in B)) \wedge \\ & \qquad (\forall x \in B\ \forall y \in A\ (y \in B) \vee \neg \mathrm{Adj}(x,y))) \\ & \quad \vee (\exists x \in V_1\ (x \in A) \wedge (\forall y \in A\ (x = y) \vee \neg (y \in V_1))) \\ & \quad \vee \dots \\ & \quad \vee (\exists x \in V_k\ (x \in A) \wedge (\forall y \in A\ (x = y) \vee \neg (y \in V_1))))\end{aligned}$$

The first two lines express that $V_1, \dots, V_k$ shall be a partition of the vertex set, and the next ones express that for any subset A of vertices, either $G[A]$ is not connected of for some i A includes exactly one element of V_i, that is: $V_1, \dots, V_k$ is a centered coloring of G. Such a centered coloring with k colors exists if and only if G has tree-depth has most k [20]. It follows that $G \models \Phi$ if and only if $\mathrm{td}(G) \leq k$. As we only check Φ on graphs with tree depth at most 2^k (given togather with a tree-decomposition easily deduced from the DFS tree) and as Φ obviously belongs to $MSOL$, there exists, according to Theorem 8.1, a linear time algorithm to check wether G satisfies Φ. □

9. Vertex separators

A celebrated theorem of Lipton and Tarjan [16] states that any planar graph has a separator of size $O(\sqrt{n})$. Alon, Seymour and Thomas [1] showed that excluding K_h as a minor ensures the existence of a separator of size at most $O(h^{3/2}\sqrt{n})$. Gilbert, Hutchinson, and Tarjan [14] further proved that graphs with genus g have a separator of size $O(\sqrt{gn})$ (this result is optimal). Plotkin et al. [23] introduced the concept of *limited-depth minor* exclusion and have shown that exclusion of small limited-depth minors implies the existence of a small separator. Precisely, they prove that any graph excluding K_h as a depth l minor has a separator of size $O(lh^2 \log n + n/l)$ hence proving that excluding a K_h minor ensures the existence of a separator of size $O(h\sqrt{n}\log n)$.

We use the following result to show that any class of graphs with sub-exponential expansion has separators of sublinear size.

Theorem 9.1 (Plotkin et al. [23]). *Given a graph with m edges and n nodes, and integers l and h, there is an $O(mn/l)$ time algorithm that will either produce a K_h-minor of depth at most $l \log n$ or will find a separator of size at most $O(n/l + 4lh^2 \log n)$.* □

Lemma 9.2. *There exists a constant C such that any graph G has a separator of size at most $C\frac{n\log n}{z}$ whenever z is an integer such that*

$$2z(\nabla_z(G)+2) \leq \sqrt{n\log n}. \tag{3}$$

Proof. Let $l = z/\log n$ and let $h = \lfloor \nabla_z(G)+2 \rfloor$. As $\nabla_z(G) \leq f(z) < h-1$, G has no K_h minor of depth at most $l\log n$. According to Theorem 9.1, G has a separator of size at most $(C/2)(n/l + 4lh^2\log n)$ for some fixed constant C, i.e. a separator of size at most $(C/2)(\frac{n\log n}{z} + 4z(\nabla_z(G)+2)^2) \leq C\frac{n\log n}{z}$. □

Theorem 9.3. *Let $\mathcal{C}$ be a class of graphs with expansion bounded by a function f such that $\log f(x) = o(x)$.*

Then the graphs in $\mathcal{C}$ have separators of size $o(n)$.

Proof. Let $g(x) = \frac{\log f(x)}{x}$. By assumption, $g(x) = o(1)$. Define $\zeta(n)$ as the greatest integer such that

$$\log f(\zeta(n)) < \frac{\log n}{3}$$

Notice that ζ is increasing and $\lim_{n\to\infty}\zeta(n) = \infty$. From the definition of $g(x)$, we deduce $\zeta(n) = \frac{\log f(\zeta(n))}{g(\zeta(n))} = \frac{\log n}{3g(\zeta(n))} = o(\log n)$. Thus $\log(2\zeta(n)(f(\zeta(n))+2)) < \frac{\log n}{3}(1+o(1))$. It follows that if n is sufficiently large (say $n > N$), $\log(2\zeta(n)(f(\zeta(n))+2)) < \frac{\log n + \log\log n}{2}$, that is: $2\zeta(n)(f(\zeta(n))+2) < \sqrt{n\log n}$. Thus if $n > N$, G has a separator of size at most $C\frac{n\log n}{\zeta(n)} = 3g(\zeta(n))n = o(n)$. □

As random cubic graphs almost surely have bisection width at least $0.101n$ (Kostochka and Melnikov, 1992), they have almost surely no separator of size smaller than $n/20$ It follows that if $\log f(x) = (\log 2)x$, the graphs have no sublinear separators any more. This shows the optimality of Theorem 9.3.

References

[1] N. Alon, P.D. Seymour, and R. Thomas, *A separator theorem for graphs with excluded minor and its applications*, Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, 1990, pp. 293–299.

[2] N. Alon, R. Yuster, and U. Zwick, *Color-coding*, J. Assoc. Comput. Mach. **42** (1995), no. 4, 844–856.

[3] H.L. Bodlaender, J.R. Gilbert, H. Hafsteinsson, and T. Kloks, *Approximating tree-width, pathwidth, frontsize, and shortest elimination tree*, Journal of Algorithms (1995), no. 18, 238–255.

[4] M. Chrobak and D. Eppstein, *Planar orientations with low out-degree and compaction of adjacency matrices*, Theoretical Computer Science **86** (1991), 243–266.

[5] D. Coppersmith and S. Winograd, *Matrix multiplication via arithmetic progressions*, J. Symbolic Comput. **9** (1990), 251–280.

[6] B. Courcelle, *Graph rewriting: an algebraic and logic approach*, Handbook of Theoretical Computer Science (J. van Leeuwen, ed.), vol. 2, Elsevier, Amsterdam, 1990, pp. 142–193.
[7] ———, *The monadic second-order logic of graphs I: recognizable sets of finite graphs*, Inform. Comput. **85** (1990), 12–75.
[8] M. DeVos, G. Ding, B. Oporowski, D.P. Sanders, B. Reed, P.D. Seymour, and D. Vertigan, *Exluding any graph as a minor allows a low tree-width 2-coloring*, Journal of Combinatorial Theory, Series B **91** (2004), 25–41.
[9] David Eppstein, *Subgraph isomorphism in planar graphs and related problems*, Proc. 6th Symp. Discrete Algorithms, ACM and SIAM, January 1995, pp. 632–640.
[10] ———, *Subgraph isomorphism in planar graphs and related problems*, J. Graph Algorithms & Applications **3** (1999), no. 3, 1–27.
[11] ———, *Diameter and treewidth in minor-closed graph families*, Algorithmica **27** (2000), 275–291, Special issue on treewidth, graph minors, and algorithms.
[12] H. de Fraysseix and P. Ossona de Mendez, *PIGALE: Public Implementation of a Graph Algorithm Library and Editor*, Free Software (GPL licence), 2002, `http://pigale.sourceforge.net`.
[13] ———, *Handbook of graph drawing and visualization*, ch. PIGALE, CRC Press, 2005.
[14] J.R. Gilbert, J.P. Hutchinson, and R.E. Tarjan, *A separator theorem for graphs of bounded genus*, J. Algorithms (1984), no. 5, 375–390.
[15] R. Halin, *S-functions for graphs*, J. Geom. **8** (1976), 171–176.
[16] R. Lipton and R.E. Tarjan, *A separator theorem for planar graphs*, SIAM Journal on Applied Mathematics **36** (1979), no. 2, 177–189.
[17] G.L. Miller, S.-H. Teng, W. Thurston, and S.A. Vavasis, *Geometric separators for finite-element meshes*, SIAM J. on Scientific Computing **19** (1998), no. 2, 364–386.
[18] J. Nešetřil and P. Ossona de Mendez, *Grad and classes with bounded expansion I. decompositions*, Tech. Report 2005-739, KAM-DIMATIA Series, 2005.
[19] ———, *The grad of a graph and classes with bounded expansion*, 7th International Colloquium on Graph Theory, 2005, accepted.
[20] ———, *Tree depth, subgraph coloring and homomorphism bounds*, European Journal of Combinatorics (2005), (in press).
[21] J. Nešetřil and S. Poljak, *Complexity of the subgraph problem*, Comment. Math. Univ. Carol. **26.2** (1985), 415–420.
[22] J. Plehn and B. Voigt, *Finding minimally weighted subgraphs*, Proc. 16th Int. Workshop Graph-Theoretic Concepts in Computer Science (Springer-Verlag, ed.), Lecture Notes in Computer Science, no. 484, 1991, pp. 18–29.
[23] S. Plotkin, S. Rao, and W.D. Smith, *Shallow excluded minors and improved graph decomposition*, 5th Symp. Discrete Algorithms, SIAM, 1994.
[24] N. Robertson and P.D. Seymour, *Graph minors. I. Excluding a forest*, J. Combin. Theory Ser. B **35** (1983), 39–61.
[25] ———, *Graph minors. XVI. Excluding a non-planar graph*, Journal of Combinatorial Theory, Series B **89** (2003), no. 1, 43–76.
[26] S.-H. Teng, *Combinatorial aspects of geometric graphs*, Computational Geometry (1998), no. 9, 277–287.
[27] K. Wagner, *Über eine Eigenschaft der Ebenen Komplexe*, Math. Ann. **114** (1937), 570–590.

Department of Applied Mathematics, and, Institute of Theoretical Computer Science (ITI), Charles University, Malostranské nám.25, 11800 Praha 1, Czech Republic
E-mail address: nesetril@kam.ms.mff.cuni.cz

Centre d'Analyse et de Mathématiques Sociales, CNRS, UMR 8557, 54 Bd Raspail, 75006 Paris, France
E-mail address: pom@ehess.fr