Color-coding *

Noga Alon †‡
Institute for Advanced Study
and Tel Aviv University

Raphael Yuster ‡
Dept. of Computer Science
Tel Aviv University

Uri Zwick ‡
Dept. of Computer Science
Tel Aviv University

March 1, 1994
Revised: March 9, 1995

Abstract

We describe a novel randomized method, the method of *color-coding* for finding simple paths and cycles of a specified length k, and other small subgraphs, within a given graph $G = (V, E)$. The randomized algorithms obtained using this method can be derandomized using families of *perfect hash functions*. Using the color-coding method we obtain, in particular, the following new results:

- For every fixed k, if a graph $G = (V, E)$ contains a simple cycle of size *exactly* k, then such a cycle can be found in either $O(V^\omega)$ expected time or $O(V^\omega \log V)$ worst-case time, where $\omega < 2.376$ is the exponent of matrix multiplication. (Here and in what follows we use V and E instead of $|V|$ and $|E|$ whenever no confusion may arise.)
- For every fixed k, if a *planar* graph $G = (V, E)$ contains a simple cycle of size *exactly* k, then such a cycle can be found in either $O(V)$ expected time or $O(V \log V)$ worst-case time. The same algorithm applies, in fact, not only to planar graphs, but to any *minor closed* family of graphs which is not the family of all graphs.
- If a graph $G = (V, E)$ contains a subgraph isomorphic to a *bounded tree-width* graph $H = (V_H, E_H)$ where $|V_H| = O(\log V)$, then such a copy of H can be found in *polynomial time*. This was not previously known even if H were just a path of length $O(\log V)$.

These results improve upon previous results of many authors. The third result resolves in the affirmative a conjecture of Papadimitriou and Yannakakis that the LOG PATH problem is in P. We can show that it is even in NC.

1 Introduction

Though the general *subgraph isomorphism* problem is NP-complete, various special cases of it are known to be *fixed parameter tractable* (see, e.g., [DF92] for a definition), and can be solved in polynomial time. In this work we introduce the *color-coding* method. Using this method we are able to solve more subcases of the subgraph isomorphism problem in polynomial time. We also obtain more efficient solutions to some subcases that already had polynomial time solutions.

---

*Work supported in part by **The basic research foundation** administrated by **The Israel academy of sciences and humanities** and by grant No. 93-6-6 of the **Sloan foundation**. A preliminary version of this paper appears in the Proceedings of the 26th STOC, 1994.

†Institute for Advanced study, school of Mathematics, Princeton, NJ 08540, USA.

‡School of Mathematical Sciences, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv 69978, ISRAEL. E-mail addresses of authors: {noga,raphy,zwick}@math.tau.ac.il.

The color-coding method is a randomized method. The vertices of the graph $G = (V, E)$ in which a subgraph isomorphic to $H = (V_H, E_H)$ is sought are randomly colored by $k = |V_H|$ colors. If $|V_H| = O(\log V)$, then with a small probability, but only polynomially small (i.e., one over a polynomial), all the vertices of a subgraph of G which is isomorphic to H, if there is such a subgraph, will be colored by distinct colors. This, as we shall see, makes the task of finding this 'color-coded' subgraph much easier.

The randomized algorithms obtained using the color-coding method are easily derandomized with only a small loss in efficiency. All that is needed to derandomize them is a family of colorings of $G = (V, E)$ so that every subset of k vertices of G is assigned distinct colors by at least one of these colorings. What is required, in other words, is a family of *perfect hash functions* from $\{1, 2, \ldots, |V|\}$ to $\{1, 2, \ldots, k\}$.

Perhaps the simplest interesting subcases of the subgraph isomorphism problem are the following: Given a directed or undirected graph $G = (V, E)$ and a number k, does G contain a simple (directed) path of length k? Does G contain a simple (directed) cycle of length *exactly* k?

We show, using color-coding, that a simple directed path of length k, in a directed graph $G = (V, E)$ that contains such a path, can be found in either $2^{O(k)} \cdot E$ expected time or in $2^{O(k)} \cdot E \log V$ or $O(k! \cdot E)$ worst-case times. If the graph $G = (V, E)$ is undirected then E in the above bounds can be replaced by V. This improves a recent $O(2^k k! \cdot V)$ worst-case bound of Bodlaender [Bod93] that applies only to undirected graphs. Note in particular that we can decide in polynomial time whether a directed or undirected graph $G = (V, E)$ contains a simple (directed) path of length $\Theta(\log V)$. This resolves in the affirmative a conjecture of Papadimitriou and Yannakakis [PY93]. The exponential dependence on k in the above bounds is probably unavoidable as the problem is NP-complete if k is part of the input.

By another application of the color-coding method, we show that a simple directed or undirected cycle of size exactly k, in a (directed) graph $G = (V, E)$ that contains such a cycle, can be found in either $2^{O(k)} \cdot VE$ or $2^{O(k)} \cdot V^\omega$ expected time or in $2^{O(k)} \cdot VE \log V$ or $2^{O(k)} \cdot V^\omega \log V$ worst-case time. This improves (in many cases) an $O(k! \cdot VE)$ worst-case bound obtained by Monien [Mon85].

For $k \leq 7$ we can *count* the number of cycles of length k in a graph $G = (V, E)$ in $O(V^\omega)$ worst-case time. This uses different techniques and will appear elsewhere. In [YZ94], it is shown that for any *even* k, cycles of length k in *undirected* graphs that contain them can be found in $O(V^2)$ worst-case time.

When applied to *planar* graphs, or to any non-trivial *minor-closed* family of graphs, the color-coding method yields optimal (in the expected case) or almost optimal (in the worst-case) algorithms for finding simple cycles of a given length. These algorithms use the fact that graphs from a non-trivial minor-closed family of graphs are of bounded *degeneracy* (see Section 5 for definition). We remind the reader that a *minor* of a graph G is any graph that can be obtained from G by removing and contracting edges. A family $\mathcal{C}$ of graphs is minor-closed if a minor of a member of $\mathcal{C}$ is also a member of $\mathcal{C}$. A family $\mathcal{C}$ is non-trivial if it does not include all the graphs. The family of planar graphs is easily seen to be such a non-trivial minor-closed family. Given a directed or undirected planar graph $G = (V, E)$ (or a graph from a non-trivial minor-closed family $\mathcal{C}$) that contains a simple (directed) cycle of size k, such a (directed) cycle can be found in $O(V)$ expected time or $O(V \log V)$ worst-case time and even in $O(V)$ worst-case time if $k \leq 5$. This improves and extends an $O(V \log V)$ worst-case bound, for $k = 5, 6$, obtained by Richards [Ric86] using planar separators and an $O(V)$ worst-case bound, for $k = 3, 4$, obtained by Chiba and Nishizeki [CN85]. Algorithms for finding triangles in planar graphs in $O(V)$ time were also obtained by Papadimitriou and Yannakakis [PY81] and Itai and Rodeh [IR78].

Our initial goal was to obtain efficient algorithms for finding simple paths and cycles in graphs. The algorithms we developed using the color-coding method turned out however to have a much wider range of applicability. The linear time (i.e., $2^{O(k)} \cdot E$ for directed graphs and $2^{O(k)} \cdot V$ for undirected graphs) bounds quoted above for simple paths apply in fact to any *forest* on k vertices. The $2^{O(k)} \cdot V^\omega$ bound quoted for simple cycles applies in fact to any *series-parallel* graph on k vertices. More generally, if $G = (V, E)$ contains a subgraph isomorphic to a graph $H = (V_H, E_H)$ whose *tree-width* is at most t, then such a subgraph can be found in $2^{O(k)} \cdot V^{t+1}$ expected time, where $k = |V_H|$. Note that forests have tree-width 1 while series-parallel graphs have tree-width 2. Our algorithm improves an algorithm of Plehn and Voigt [PV90] that has a running time of $k^{O(k)} \cdot V^{t+1}$. Our result gives, as far as we know, the most general subcase of the subgraph isomorphism problem known to be solvable in polynomial time.

The concept of tree-width was introduced by Robertson and Seymour (see e.g., [RS86a]) in their series of works on graph minors. Robertson and Seymour use this concept, together with other ingredients, some of them non-constructive, to show that the *subgraph homomorphism* problem (given a graph $G = (V, E)$ and a graph $H = (V_H, E_H)$, does G have a subgraph homomorphic to H?) and the *minor containment* problem (given a graph $G = (V, E)$ and a graph $H = (V_H, E_H)$, does G have H as a minor?) can be solved in $O(V^3)$ time for every fixed H and even in $O(V)$ time if H is a path (see [Joh87] for a survey of these results). A graph contains a simple path of length k, as a subgraph, if and only if it contains such a path as a minor. This gives therefore an alternative $O(V)$ time algorithm for deciding whether an undirected graph contains a simple path of length k. The obtained algorithm has however a worse dependence on k.

Fürer and Raghavachari [FR92] and Karger et al. [KMR93] give algorithms for finding, in polynomial time, a simple path of length $\log_2 V$ in Hamiltonian or weakly undirected Hamiltonian graphs. Our results significantly extend this result as we can find, in polynomial time, a simple path of length $c \log_2 V$, for any fixed $c > 0$, in any directed or undirected graph that contains such a path.

In the next section we describe a simplified version of the color-coding method called the method of *random orientations*. The basic color-coding method is described in Section 3. In Section 4 we show how to derandomize the algorithms obtained using these methods. More sophisticated applications of the color-coding method are described in Sections 5 and 6.

2 Random Orientations

Let $G = (V, E)$ be an undirected graph. Suppose we want to find pairs of vertices connected by *simple* paths of length *exactly* k. By raising the adjacency matrix $A = A_G$ of $G = (V, E)$ to the k-th power, or by using other methods, we can easily find all pairs of vertices connected by paths of length k. Most of these paths, however, would not be simple. How can we weed out the non-simple paths? An easy way of doing this is by choosing a random *acyclic* orientation of the graph G. Such an orientation is obtained by choosing a random permutation $\pi : V \to \{1, \ldots, |V|\}$ and directing an edge $(u, v) \in E$ from u to v if and only if $\pi(u) < \pi(v)$. Denote the resulting directed graph by $\vec{G}$. Every directed path of length k in $\vec{G}$ is simple and corresponds to a simple path of length k in G. Every simple path of length k in G, on the other hand, has a $2/(k+1)!$ chance of becoming a directed path (in either direction) in $\vec{G}$. This simple observation yields the following two results:

Theorem 2.1 *A simple directed or undirected path of length k in a (directed or undirected) graph $G = (V, E)$ that contains such a path can be found in $O((k+2)! \cdot V)$ expected time in the undirected case and in $O((k+1)! \cdot E)$ expected time in the directed case.*

Proof : Consider first the undirected case. An algorithm with $O((k+1)! \cdot E)$ expected time is immediate. Simply choose a random acyclically oriented version $\vec{G}$ of G and find the longest directed path in it. This can easily be done in $O(E)$ time (see e.g., [CLR90], p. 538). The longest path in $\vec{G}$ would be of length at least k with a probability of at least $2/(k+1)!$. If the longest path is of length less than k, repeat the process. The expected number of times this process is repeated before the desired path is found is at most $(k+1)!/2$.

To reduce the $O((k+1)! \cdot E)$ complexity to the desired $O((k+2)! \cdot V)$ we use the well known (and easy) fact that every graph with V vertices and at least $k|V|$ edges contains a path of length k. The known proofs of this fact easily supply a method of finding such a path in $O(k \cdot V)$ time. A specific way of incorporating this into our algorithm follows an idea of Bodlaender [Bod93]. We start a DFS (depth-first search) on the graph. If a vertex of depth k is ever found, we stop and output the path from the root to this vertex. If no such vertex is found, the graph contains at most $k|V|$ edges (as all back-edges point to ancestors) and we may apply the algorithm described above.

The random orientation method can be easily adapted to work for directed graphs. To turn a directed graph $G = (V, E)$ into an acyclic graph we again choose a random permutation $\pi : V \to \{1, \ldots, |V|\}$ and then *delete* every edge $(u, v) \in E$ for which $\pi(u) > \pi(v)$. The $O((k+1)! \cdot E)$ expected time algorithm described above works also in this case. $\square$

Theorem 2.2 *A simple directed or undirected cycle of length k in a (directed or undirected) graph $G = (V, E)$ that contains such a cycle can be found in $O(k! \log k \cdot V^\omega)$ expected time.*

Proof : The algorithm used is very similar to the one used in the proof of Theorem 2.1. We choose a random acyclically oriented version $\vec{G}$ of G. We now raise the adjacency matrix of $\vec{G}$ to the $k-1$-st power using $O(\log k)$ matrix multiplications. This gives us all pairs of vertices connected by directed paths of length $k-1$ in $\vec{G}$. If the vertices in any one of these pairs are connected by an edge, a k-cycle is found. This will happen with a probability of at least $2/k!$. This process is repeated an expected number of at most $k!/2$ times. A very similar algorithm can be used to find directed simple cycles in directed graphs. $\square$

One can match the performance of the algorithm described in Theorem 2.1 using a deterministic algorithm by combining techniques of Monien [Mon85] and Bodlaender [Bod93]. The obtained algorithm works in $O(k! \cdot E)$ time for directed graphs or in $O(k! \cdot V)$ time for undirected graphs. As this algorithm does not use the color-coding method we omit its description.

We note that although the $O(V^\omega)$ algorithm of Theorem 2.2 is extremely simple, no such result was previously known. This algorithm is derandomized in Section 4 yielding an $O(V^\omega \log V)$ deterministic version. For $k \le 7$ we can use other methods to count the number of cycles of length k in deterministic time $O(V^\omega)$. This will appear elsewhere.

The $k!$ dependence on k in Theorems 2.1 and 2.2 is improved in the next section.

3 Random colorings

Let $G = (V, E)$ be a directed or undirected graph. Consider again the problem of finding simple directed or undirected paths of length $k-1$ in G. Choose a random coloring of the vertices of G with k colors. A path in G is said to be *colorful* if each vertex on it is colored by a distinct color. A colorful path in G is clearly simple. Each simple path of length $k-1$, on the other hand, has a chance of $k!/k^k > e^{-k}$ to become colorful. Note that this is only exponentially small in k. How much time is needed to find a colorful path of length $k-1$ in G, if one exists, or all pairs of vertices connected by colorful paths of length $k-1$ in G? The next lemmas give answers to these questions.

Lemma 3.1 *Let $G = (V, E)$ be a directed or undirected graph and let $c : V \to \{1, \ldots, k\}$ be a coloring of its vertices with k colors. A colorful path of length $k-1$ in G, if one exists, can be found in $2^{O(k)} \cdot E$ worst-case time.*

Proof : We describe at first an $2^{O(k)} \cdot E$ time algorithm that receives as input the graph $G = (V, E)$, the coloring $c : V \to \{1, \ldots, k\}$ and a vertex $s \in V$, and finds a colorful path of length $k-1$ that starts at s, if one exists. To find a colorful path of length $k-1$ in G that starts somewhere, we just add a new vertex s' to V, color it with a new color 0 and connect it with edges to all the vertices of V. We now look for a colorful path of length k that starts at s'.

A colorful path of length $k-1$ that starts at some specified vertex s is found using a dynamic programming approach. Suppose we already found for each vertex $v \in V$ the possible sets of colors on colorful paths of length i that connect s and v. Note that we do not record all colorful paths connecting s and v, we only record the color sets appearing on such paths. For each vertex v we have therefore a collection of at most $\binom{k}{i}$ color sets. We inspect every subset C that belongs to the collection of v, and every edge $(v, u) \in E$. If $c(u) \notin C$, we add the set $C \cup \{c(u)\}$ to the collection of u that corresponds to colorful paths of length $i+1$. The graph G contains a colorful path of length $k-1$ with respect to the coloring c if and only if the final collection, that corresponding to paths of length $k-1$, of at least one vertex is non-empty. The number of operations performed by the algorithm outlined is at most $O(\sum_{i=0}^{k} i\binom{k}{i} \cdot |E|)$ which is clearly $O(k2^k \cdot E)$. $\square$

Lemma 3.2 *Let $G = (V, E)$ be a directed or undirected graph and let $c : V \to \{1, \ldots, k\}$ be a coloring of its vertices with k colors. All pairs of vertices connected by colorful paths of length $k-1$ in G can be found in either $2^{O(k)} \cdot VE$ or $2^{O(k)} \cdot V^\omega$ worst-case time.*

Proof : The $2^{O(k)} \cdot VE$ algorithm is obtained by simply running the algorithm described in the proof of the previous Lemma $|V|$ times, once for each starting vertex.

To obtain the $2^{O(k)} \cdot V^\omega$ algorithm we use the following recursive approach. Enumerate all partitions of the color set $\{1, 2, \ldots, k\}$ into two subsets C_1, C_2 of size $k/2$ each (to simplify the presentation we omit floor and ceiling signs). There are only $\binom{k}{k/2} < 2^k$ such partitions. For each such partition C_1, C_2, let V_1 be the set of vertices of G colored by colors from C_1 and V_2 be the set of vertices of G colored by colors from C_2. Let G_1 and G_2 be the subgraphs of G induced by V_1 and V_2 respectively. Recursively find all pairs of vertices in G_1 and in G_2 connected by colorful paths of length $k/2 - 1$. Collect this information into two Boolean matrices A_1 and A_2. Let B be a Boolean matrix that describes the adjacency relations between the vertices of V_1 and those of V_2. The Boolean product $A_1 B A_2$ gives all pairs of vertices in V that are connected by colorful paths of length exactly $k-1$, where the first $k/2$ vertices on the paths are colored by colors from C_1 and the last $k/2$ vertices are colored by colors from C_2. By OR-ing the matrices obtained for all the partitions we obtain the desired result. It is easy to verify that the complexity of this approach is indeed $2^{O(k)} \cdot V^\omega$, as the number of matrix multiplication used, $t(k)$, satisfies the recurrence $t(k) \leq 2^k t(k/2)$. $\square$

The $2^{O(k)} \cdot V^\omega$ algorithm outlined above finds all pairs of vertices connected by colorful paths of length $k-1$. To find the colorful paths themselves we can use an algorithm by Alon and Naor [AN94] for finding *witnesses* for Boolean matrix multiplication. We omit the details.

Using the above Lemmas we immediately get the following results.

Theorem 3.3 *A simple directed or undirected path of length $k-1$ in a (directed or undirected) graph $G = (V, E)$ that contains such a path can be found in $2^{O(k)} \cdot V$ expected time in the undirected case and in $2^{O(k)} \cdot E$ expected time in the directed case.*

Theorem 3.4 *A simple directed or undirected cycle of size k in a (directed or undirected) graph $G = (V, E)$ that contains such a cycle can be found in either $2^{O(k)} \cdot VE$ or $2^{O(k)} \cdot V^\omega$ expected time.*

In [YZ94], the last two authors show that a simple cycle of size k, where k is *even*, in an *undirected* graph that contains such a cycle can be found *deterministically* in $O(k! \cdot V^2)$ time.

As mentioned in the introduction, the color-coding method can be used to efficiently find not only paths and cycles but any subgraph with a bounded tree-width. This generalization is briefly presented in Section 6.

4 Derandomized orientations and colorings

The randomized algorithms of the previous two sections can be derandomized with only a small loss of efficiency. The $2^{O(k)}$ dependence on k is retained for the small price of an extra $\log V$ factor to the complexity.

What we need, if we want to give every simple path of length, say, $k-1$ in a graph $G=(V,E)$ a chance of being discovered, is a list of colorings of V such that for every subset $V' \subseteq V$ of size $|V'|=k$ there exists a coloring in the list that gives each vertex in V' a distinct color. What we need, in other words, is a k-perfect family of *hash functions* from $\{1,2,\ldots,|V|\}$ to $\{1,2,\ldots,k\}$.

Schmidt and Siegal [SS90], following Fredman, Komlós and Szemerédi [FKS84], give an explicit construction of a k-perfect family from $\{1,2,\ldots,n\}$ to $\{1,2,\ldots,k\}$ in which each function is specified using $O(k)+2\log\log n$ bits. The size of the family is therefore $2^{O(k)}\log^2 n$. The value of each one of these functions on each specified element of $\{1,2,\ldots,n\}$ can be evaluated in $O(1)$ time. Using this family we can derandomize the algorithms presented in the previous sections but the incurred cost would be a multiplicative factor of $\log^2 V$ and not of $\log V$ as promised.

As pointed out by Moni Naor, the size of the desired family of hash functions can be reduced to $2^{O(k)}\log n$ in the following way. First construct a k-perfect family that maps $\{1,2,\ldots,n\}$ to $\{1,2,\ldots,k^2\}$. Next construct a k-perfect family that maps $\{1,2,\ldots,k^2\}$ to $\{1,2,\ldots,k\}$. The desired family is obtained by composing these two families of hash functions. A k-perfect family of size $2^{O(k)}$ from $\{1,2,\ldots,k^2\}$ to $\{1,2,\ldots,k\}$ can be obtained using the construction of Schmidt and Siegal [SS90].

To construct a k-perfect family of size $k^{O(1)}\log n$ from $\{1,2,\ldots,n\}$ to $\{1,2,\ldots,k^2\}$ we use small probability spaces that support sequences of *almost ℓ-wise independent* random variables. A sequence $X_1,\ldots,X_n$ of random Boolean variables is (ε,ℓ)-independent if for any ℓ positions $i_1<i_2<\cdots<i_\ell$ and any ℓ bits $\alpha_1,\ldots,\alpha_\ell$ we have

$$\left|\Pr[X_{i_1}=\alpha_1,\ldots,X_{i_\ell}=\alpha_\ell]-2^{-\ell}\right|<\varepsilon \quad .$$

Note in particular that if the sequence $X_1,\ldots,X_n$ is $(2^{-\ell},\ell)$-independent, then any subset of ℓ variables attains each one of its 2^ℓ possible values with some positive probability.

Constructions of small probability spaces that admit almost ℓ-wise independent random variables were obtained by Naor and Naor [NN90] and Alon et al. ([AGHP92], [ABN+92]). The size of sample spaces that support n random variables that are (ε,ℓ)-independent can be as small as $2^{O(\ell+\log\frac{1}{\varepsilon})}\log n$ and they can be constructed in $2^{O(\ell+\log\frac{1}{\varepsilon})}n\log n$ time.

To construct a k-perfect family of size $k^{O(1)}\log n$ from $\{1,2,\ldots,n\}$ to $\{1,2,\ldots,k^2\}$ we use a probability space of size $k^{O(1)}\log n$ that supports ℓn random variables that are $(2^{-2\ell},2\ell)$-independent, where $\ell=2\log k$. We attach ℓ random variables to each element of $\{1,2,\ldots,n\}$ thereby assigning it a color from $\{1,2,\ldots,k^2\}$. Consider two elements $1\le i<j\le n$. The probability that i and j are assigned the same color is at most $2^{1-\ell}=2/k^2$. The probability that two distinct elements from $\{1,2,\ldots,n\}$ are assigned the same color is therefore strictly less than 1 and the obtained family is indeed k-perfect.

The algorithms obtained using random orientations can also be derandomized. Instead of choosing a random permutation $\pi: V\to\{1,\ldots,|V|\}$, choose a random coloring $c: V\to\{1,\ldots,k\}$ and remove all edges $(u,v)\in E$ such that $c(v)\ne c(u)+1$. An edge $(u,v)\in E$ such that $c(v)=c(u)+1$ will be directed from u to v. The obtained graph $\vec{G}$ is again acyclic. Each simple path of length k in G now has a probability of $2k^{-k}$ of becoming a directed path in $\vec{G}$. Note the difference between the color-coding method and this version of the random orientations method. Here we require the first vertex on the path to be colored by 1, the second by 2 and so forth. In the color-coding method we just require the vertices on the path to be colored by the distinct colors, in *some* order.

A list of colorings in which each sequence $v_1,\ldots,v_k$ of k vertices from V is colored consecutively by

$1, 2, \ldots, k$ by at least one coloring of the list is easily obtained by using sequences of $(k \log k) \cdot n$ random variables that are almost $k \log k$-wise independent. The size of the list will be $k^{O(k)} \log V$. Such a list is used for derandomization purposes in Section 5.

5 Finding cycles in minor-closed families of graphs

An undirected graph $G = (V, E)$ is *d-degenerate* (see Bollobás [Bol78], p. 222) if there exists an acyclic orientation of it in which $d_{out}(v) \leq d$ for every $v \in V$. The smallest d for which G is d-degenerate is called the *degeneracy* or the *max-min degree* of G and is denoted by $d(G)$. It can be easily seen (see again [Bol78]) that $d(G)$ is the maximum over the minimum degrees of all the subgraphs of G. Clearly, if G is d-degenerate then $|E| \leq d \cdot |V|$. The following simple lemma, whose proof is omitted, is part of the folklore (see, e.g., [MB83]).

Lemma 5.1 *Let $G = (V, E)$ be a connected undirected graph $G = (V, E)$. An acyclic orientation of G such that for every $v \in V$ we have $d_{out}(v) \leq d(G)$ can be found in $O(E)$ time.*

Let G be an undirected graph. A graph H is a *minor* of G if it can be obtained from G by the removal and the contraction of edges. A family $\mathcal{C}$ of graphs is said to be *minor-closed* if a minor of a graph of the family is also a member of the family.

It is known (see [Bol86], p. 7) that if $\mathcal{C}$ is a non-trivial minor-closed family of graphs, i.e., a minor-closed family which is not the family of all graphs, then all graphs in $\mathcal{C}$ are of bounded degeneracy. In other words, there exists a constant $d = d_{\mathcal{C}}$ such that every $G \in \mathcal{C}$ satisfies $d(G) \leq d$. As an example, consider the family of *planar graphs*. It is minor-closed and the degeneracy of every planar graph is at most 5 (as each planar graph has a vertex whose degree is at most 5). The second result claimed in the abstract follows therefore from the following theorem and its derandomized version.

Theorem 5.2 *Let $\mathcal{C}$ be a non-trivial minor-closed family of graphs and let $k \geq 3$ be a fixed integer. Then, there exists a randomized algorithm that given a graph $G = (V, E)$ from $\mathcal{C}$, finds a C_k (a simple cycle of size k) in G, if one exists, in $O(V)$ expected time.*

Proof : Let $G = (V, E)$ be a graph from $\mathcal{C}$ that contains a C_k. Choose a random coloring $c : V \to \{1, \ldots, k\}$ of the vertices of G. A C_k in G is said to be *well-colored* if the vertices on it are consecutively colored by $1, 2, \ldots, k$. With a probability of $2/k^{k-1}$, the C_k present in G will be well-colored.

We describe a randomized algorithm, whose running time is $O(k \cdot V)$, that given a graph $G = (V, E)$ from a minor-closed family of degeneracy $d = O(1)$ and given a coloring $c : V \to \{1, \ldots, k\}$, has a probability of at least $1/(2d)^k$ of finding a well-colored C_k in G, if one exists. By combining this algorithm with the initial random coloring phase, we obtain an $O(k \cdot V)$ time algorithm that finds a C_k in G, if one exists, with a probability of at least $\frac{2}{(2d)^k k^{k-1}}$. This may be a very small probability but it depends only on k and d and not on the size of the graph. By rerunning the algorithm, with an independent set of choices, in case of failure, we obtain an $O((2dk)^k \cdot V)$ expected time algorithm for finding a C_k in graphs that contain one.

We assume that all the edges of G connect vertices that are colored by consecutive colors (modulo k). Edges violating this property can be removed as this will not remove any well-colored C_k from G. The randomized algorithm for finding a well-colored C_k in G starts by orienting G so that the out-degree of every vertex is at most d. The edges that leave a vertex $v \in V$ are assigned the indices $1, \ldots, d_{out}(v) \leq d$ in an arbitrary manner. This takes only $O(V)$ time. The well-colored C_k assumed to exist in G contains an edge between a vertex v_{k-1} colored by $k-1$ and a vertex v_k colored by k. The algorithm guesses, by flipping fair coins, the orientation and the index of this edge. There are two possible orientations and d possible indices. If the guess is that the edge is directed from v_{k-1} to v_k and the guessed index is i, then all edges that leave vertices colored by $k-1$ but whose index is not i are removed from the graph. If the opposite direction is guessed, then the same is done with edges that leave vertices colored by k. The resulting graph, which we denote by G', still contains a well-colored C_k with a probability of at least $1/2d$.

The subgraph of G' induced by vertices colored by $k-1$ and k is a forest of rooted stars. We contract each such star into a single vertex and assign each such new vertex the color $k-1$. We obtain a new graph G'' and a new coloring c''. It is easy to see that G'' contains a well-colored C_{k-1} if and only if G' contains a well-colored C_k. To verify this, recall that each edge of G', and therefore also each edge of G'', connects consecutively colored vertices. As G'' is a minor of G and as $\mathcal{C}$ is minor-closed, G'' is also a member of $\mathcal{C}$.

We now apply the algorithm recursively and look for a well-colored C_{k-1} in G''. This takes $O((k-1)\cdot V)$ time and yields a well-colored C_{k-1} with a probability of at least $1/(2d)^{k-1}$. If a well-colored C_{k-1} is found in G'', a well-colored C_k in G is easily reconstructed. The overall time spent is $O(k\cdot V)$ and the probability of finding a well-colored C_k is at least $1/(2d)^k$.

To complete the picture, we have to specify the way in which the recursion bottoms. The contractions used in the various stages of the algorithm generate self-loops, which are immediately removed. Parallel edges, however, may only occur when contractions are applied to a graph colored by three colors and a two-colored graph is obtained. A well-colored C_2 in such a graph is just a pair of parallel edges and such a pair, if one exists, can be easily found in $O(V)$ time.

Alternatively, we can stop the recursion when $k=3$ and use an existing $O(E\cdot d(G))$ time algorithm (see [CN85]) for finding triangles (C_3's) in a general graph $G=(V,E)$. Note that any triangle in a three-colored graph is well-colored and that $O(E\cdot d(G))$ is $O(V)$ in our case. □

The algorithm just described can again be derandomized.

Theorem 5.3 *Let $\mathcal{C}$ be a non-trivial minor-closed family of graphs and let $k\geq 3$ be a fixed integer. There exists a deterministic algorithm that decides whether a given graph $G=(V,E)$ from $\mathcal{C}$ contains a C_k, and finds one, if one exists, in $O(V\log V)$ worst-case time.*

Proof : We derandomize the algorithm given in the proof of Theorem 5.2. Instead of using random colorings, we exhaust a list of $k^{O(k)}\log V$ colorings that has the property that every sequence $v_1,\ldots,v_k$ of k vertices from V is consecutively colored by $1,2,\ldots,k$ in at least one coloring of the list. Such a list was also used for derandomizing the algorithms obtained using the random orientations method. Instead of guessing the direction and index of each edge in the well-colored C_k, we exhaust, for each coloring, all the $(2d)^k$ possible choices. If G contains a C_k then at least one C_k will be found in this way. □

Theorems 5.2 and 5.3 deal with undirected graphs. With only minor modifications they can be used however to find directed cycles in directed graphs whose undirected versions belong to any nontrivial minor-closed family $\mathcal{C}$.

Without using the color-coding method, we have obtained in [AYZ94] the following result.

Theorem 5.4 *Let $G=(V,E)$ be directed or undirected graph. A (directed or undirected) C_5 in G, if one exists, can be found in $O(E\cdot(d(G))^2)$ worst-case time.*

As a corollary, we get that if $\mathcal{C}$ is a non-trivial minor-closed family of graphs and $G=(V,E)$ is a member of $\mathcal{C}$, then a C_5 in a G, if one exists, can be found in $O(V)$ time.

Eppstein [Epp95] showed recently that if $G=(V,E)$ is a *planar* graph and H is a graph on k vertices, then a copy of H in G, if one exists, can be found in $O(k^{O(k)}V)$ time. Eppstein's result also applies to graphs of a bounded genus but it does not apply, like our method, to all minor-closed families of graphs.

6 Finding bounded tree-width subgraphs

A slight modification of the method used in Section 3 to find paths can be used to find any fixed directed or undirected forest. (A directed forest is a directed graph whose undirected version is a forest.)

Theorem 6.1 *Let F be a directed or undirected forest on k vertices. Let $G = (V, E)$ be a directed or undirected graph. A subgraph of G isomorphic to F, if one exists, can be found in $2^{O(k)} \cdot E$ expected time in the directed case, and in $2^{O(k)} \cdot V$ expected time in the undirected case.*

Proof : We start, as usual, by choosing a random coloring $c : V \to \{1, \ldots, k\}$ of the graph G, which is assumed to contain a copy of F. With a probability of at least e^{-k}, the copy of F in G will become colorful, i.e., each vertex in it will get a different color. Assume this is the case. Suppose that F is composed of ℓ (directed) trees $T_1, \ldots, T_\ell$ with $k_1, \ldots, k_\ell$ vertices each. Let F_i, for $1 \le i \le \ell$ be the (directed) forest composed of $T_1, \ldots, T_i$. We find, for each $1 \le i \le \ell$, the color sets that appear on colorful copies of T_i in G. It is then easy to find, in $2^{O(k)}$ time, the color sets that appear on colorful copies F_i, for $1 \le i \le \ell$. Note that copies of T_i and T_j, for $i \ne j$, with disjoint color sets are necessarily disjoint. If the collection corresponding to $F = F_\ell$ is not empty, then G contains a colorful copy of F. Such a copy is found if with every color set found we keep at least one copy of a corresponding subgraph colored by it.

Let $T = T_i$ be a (directed) tree on $m = k_i$ vertices, where $1 \le i \le \ell$, and let r be an arbitrary vertex in T. In $2^{O(m)} \cdot E$ time, we find, for each vertex $v \in V$, all color sets that appear on colorful copies of T in G in which v plays the role of r. If T contains only a single vertex, this is easily done. Otherwise, let $e_T = (r, r')$ be a (directed) edge in T. The removal of e_T from T breaks T into two (directed) subtrees T' and T''. We recursively find, for each vertex $v \in V$, the color sets that appear on colorful copies of T' in which v plays the role of r, and the color sets that appear on colorful copies of T'' in which v plays the role of r'. For every (directed) edge $e = (u, v) \in E$, if C' is a color set that appears in u's collection (corresponding to T'), if C'' is a color set that appears in v's collection (corresponding to T''), and if $C' \cap C'' = \emptyset$, then $C' \cup C''$ is added to the collection of u (corresponding to T). It is easy to see that the complexity of this recursive algorithm is $2^{O(m)} \cdot E$, as required.

To obtain the better bound in the undirected case, we use the fact a graph $G = (V, E)$ with at least $k \cdot |V|$ edges contains, as a subgraph, any forest on k vertices. □

The algorithms in the last theorem can obviously be derandomized using the techniques described in Section 4.

The basic ideas used in the above proof can be used to obtain an algorithm that looks not only for trees and forests but for any graph with a bounded tree-width, a notion introduced by Robertson and Seymour [RS86a].

Definition 6.2 *A* tree-decomposition *of a graph $G = (V, E)$ is a pair (X, T) where $T = (I, F)$ is a tree and $X = \{X_i : i \in I\}$ is a family of subsets of V such that (i) $\cup_{i \in I} X_i = V$; (ii) for every edge $(u, v) \in E$, there exists an $i \in I$ such that $u, v \in X_i$; and (iii) if $i, j, k \in I$ and j is on the path from i to k in T, then $X_i \cap X_k \subseteq X_j$. The* tree-width *of the tree-decomposition (X, T) is $\max_{i \in I} |X_i| - 1$. The tree-width of a graph G is the minimum tree-width over all possible tree-decompositions of G. Graphs with tree-width at most t are also called* partial t-trees.

The proof of the following result is similar to that of Theorem 6.1 and is thus omitted.

Theorem 6.3 *Let H be a directed or undirected graph on k vertices with tree-width t. Let $G = (V, E)$ be a (directed or undirected) graph. A subgraph of G isomorphic to H, if one exists, can be found in $2^{O(k)} \cdot V^{t+1}$ expected time and in $2^{O(k)} \cdot V^{t+1} \log V$ worst-case time.*

A result similar to the above result, but with a worse dependence on k, was obtained by Plehn and Voigt [PV90]. If a real weight function $\beta : E \to R$ is defined on the edges of G, then the algorithm of Plehn and Voigt, as well as our deterministic algorithm, can be adapted to find the copy of H in G with the minimal/maximal total weight.

As a very special case of Theorem 6.3, we get that the LOG PATH problem is in P. It is not difficult to check that all the algorithms we have described are easily parallelizable. It follows therefore that the LOG PATH problem is even in NC.

As mentioned in the introduction, Robertson and Seymour [RS86b] showed that if $\mathcal{C}$ is a minor closed family of graphs that excludes at least one planar graph H, then there exists a (huge) constant c_H such that every graph in $\mathcal{C}$ has tree-width at most c_H. As a simple corollary to Theorem 6.3, we get that if $G = (V, E)$ and $H = (V_H, E_H)$ such that $|V_H| = O(\log V)$ and H is, say, K_4-free (i.e., has no K_4 minor), then we can decide in polynomial time whether G contains a subgraph isomorphic to H.

7 Concluding remarks and open problems

The color-coding method is a good example for demonstrating the use of derandomization techniques. All the algorithms obtained here using this method can be easily parallelized, yielding efficient NC algorithms to the corresponding problems.

Several problems, listed below, remain open.

- Is there a polynomial time (deterministic or randomized) algorithm for deciding if a given graph $G = (V, E)$ contains a path of length, say, $\log^2 V$?
- Can the $\log V$ factor appearing in our derandomization be omitted? (This will supply, in particular, for every fixed $k \geq 3$, an optimal, linear time deterministic algorithm for deciding if a given planar graph contains a cycle of length k.)
- Is the problem of deciding whether a given graph $G = (V, E)$ contains a triangle as difficult as the Boolean multiplication of two V by V matrices?

Acknowledgment

We would like to thank Moni Naor for helpful conversations.

References

[ABN+92] N. Alon, J. Bruck, J. Naor, M. Naor, and R.M. Roth. Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs. *IEEE Transactions on Information Theory*, 38(2):509–516, 1992.

[AGHP92] N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures and Algorithms*, 3(3):289–304, 1992. Addendum in *Random Structures and Algorithms*, 4(1):119–120, 1993.

[AN94] N. Alon and M. Naor. Derandomization, witnesses for boolean matrix multiplication and construction of perfect hash functions. To appear in *Algorithmica*, 1994.

[AYZ94] N. Alon, R. Yuster, and U. Zwick. Finding and counting given length cycles. In *Proceedings of the 2nd European Symposium on Algorithms, Utrecht, The Netherlands*, pages 354–364, 1994.

[Bod93] H.L. Bodlaender. On linear time minor tests with depth-first search. *Journal of Algorithms*, 14:1–23, 1993.

[Bol78] B. Bollobás. *Extremal graph theory.* Academic Press, 1978.

[Bol86] B. Bollobás. *Extremal graph theory with emphasis on probabilistic methods.* Regional conference series in mathematics, No. 62, American Mathematical Society, 1986.

[CLR90] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to algorithms.* The MIT Press, 1990.

[CN85] N. Chiba and L. Nishizeki. Arboricity and subgraph listing algorithms. *SIAM Journal on Computing*, 14:210–223, 1985.

[DF92] R.G. Downey and M.R. Fellows. Fixed-parameter intractability. In *Proceedings of the 7th Annual Symposium on Structure in Complexity Theory*, pages 36–49, 1992.

[Epp95] D. Eppstein. Subgraph isomorphism in planar graphs and related problems. In *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, California*, pages 632–640, 1995.

[FKS84] M.L. Fredman, J. Komlós, and E. Szemerédi. Storing a sparse table with $O(1)$ worst case access time. *Journal of the ACM*, 31:538–544, 1984.

[FR92] M. Fürer and B. Raghavachari. Approximating the minimum degree spanning tree to within one from the optimal degree. In *Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms, Orlando, Florida*, pages 317–324, 1992.

[IR78] A. Itai and M. Rodeh. Finding a minimum circuit in a graph. *SIAM Journal on Computing*, 7:413–423, 1978.

[Joh87] D.S. Johnson. The NP-completeness column: An ongoing guide. The many faces of polynomial time. *Journal of Algorithms*, 8:285–303, 1987.

[KMR93] D. Karger, R. Motwani, and G.D.S. Ramkumar. On approximating the longest path in a graph. In *Proceedings of the Workshop on algorithms and data structures, Montreal, Canada*, pages 421–432, 1993.

[MB83] D.W. Matula and L.L. Beck. Smallest-last ordering and clustering and graph coloring algorithms. *Journal of the ACM*, 30:417–427, 1983.

[Mon85] B. Monien. How to find long paths efficiently. *Annals of Discrete Mathematics*, 25:239–254, 1985.

[NN90] J. Naor and M. Naor. Small-bias probability spaces: efficient constructions and applications. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 213–223, 1990.

[PV90] J. Plehn and B. Voigt. Finding minimally weighted subgraphs. In *Proc. 16th Workshop on graph theoretic concepts in computer Science*, Lecture Notes in Computer Science, Vol. 484, pages 18–29. Springer, 1990.

[PY81] C.H. Papadimitriou and M. Yannakakis. The clique problem for planar graphs. *Information Processing Letters*, 13:131–133, 1981.

[PY93] C.H. Papadimitriou and M. Yannakakis. On limited nondeterminism and the complexity of the V-C dimension. In *Proceedings of the 8th Annual Symposium on Structure in Complexity Theory, San Diego, California*, pages 12–18, 1993.

[Ric86] D. Richards. Finding short cycles in a planar graph using separators. *Journal of Algorithms*, 7:382–394, 1986.

[RS86a] N. Robertson and P. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7:309–322, 1986.

[RS86b] N. Robertson and P. Seymour. Graph minors. V. Excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 41:92–114, 1986.

[SS90] J.P. Schmidt and A. Siegel. The spatial complexity of oblivious k-probe hash functions. *SIAM Journal on Computing*, 19(5):775–786, 1990.

[YZ94] R. Yuster and U. Zwick. Finding even cycles even faster. In *Proceedings of the 21st International Colloquium on Automata, Languages and Programming, Jerusalem, Israel*, Lecture Notes in Computer Science, Vol. 820, pages 532–543. Springer, 1994.