Coloring k–colorable graphs in constant expected parallel time

Luděk Kučera

MPI–I–93–110 February 1993

Coloring k–colorable graphs in constant expected parallel time

Luděk Kučera

MPI–I–93–110 February 1993

Coloring k-colorable graphs in constant expected parallel time

Luděk Kučera
Charles University,
Prague, Czechoslovakia [*]
and
Max Planck Institute for Computer Science,
Saarbrücken, Germany

March 9, 1993

Abstract

A parallel (CRCW PRAM) algorithm is given to find a k-coloring of a graph randomly drawn from the family of k-colorable graphs with n vertices, where $k = \log^{O(1)} n$. The average running time of the algorithm is constant, *and the number of processors is equal to $|V| + |E|$, where $|V|$, $|E|$, resp. is the number of vertices, edges, resp. of the input graph.*

Introduction

The graph coloring problem is NP-complete, and it has been proved that finding a good approximation solution is as difficult as computing of the optimal one [10]. The complexity classes R and BPP, representing practical use of randomization, do not seem to contain NP-complete problems. Therefore the only way to cope with coloring-type

[*]This research was partially supported by EC Cooperative Action IC-1000 (project ALTEC: *Algorithms for Future Technologies*)

problems is presently a probabilistic analysis of behavior of (usually deterministic) polynomial time algorithms, applied to random inputs.

The most common way of generating random input graphs is represented by the random graph $\mathcal{G}_n$, which gives all graphs with n vertices equaly likely, or by a more general model $\mathcal{G}_{n,p}$, where p is a parameter chosen usually so that sparser graphs are more likely, but all graphs with the same number of edges are generated with the same probability (the distribution $\mathcal{G}_n$ is equal to $\mathcal{G}_{n,1/2}$). Throughout the paper, we will suppose for simplicity that p is a constant, but most results can easily be generalized. We will denote $1/(1-p)$ by b.

Random graphs $\mathcal{G}_{n,p}$ do not represent an interesting tool for testing heuristic coloring algorithms. It is known that $\chi(\mathcal{G}_{n,p}) = (1 + o(1))\frac{n}{2\log_b n}$ almost surely (for the lower bound see [11], for the upper bound [3, 18]). On the other hand, it is easy to show that perhaps the simplest coloring algorithm, the greedy one, uses almost surely $(1+o(1))\frac{n}{\log_b n}$ colors when applied to $\mathcal{G}_{n,p}$, i.e. the "ratio of optimality" is almost surely $2 + o(1)$. It is interesting that even the most sophisticated among known polynomial graph coloring algorithms seem or are proved to behave in essentially the same way as the greedy one, using almost surely about $2\chi(G)$ colors. (There is however an indication that the problem of (near) optimal coloring of $\mathcal{G}_{n,p}$ might be easier that NP-hard problems (or problems hard "on average" in the sense of Levin [17]), because searching of all independent subsets of size $O(\log n)$ finds a near optimal coloring almost surely in time $n^{O(\log n)}$).

The greedy algorithm can also be parallelized so that it achieves essentially the same results in polylogarithmic time [9, 7].

All this implies that all known polynomial time algorithms color the random graphs $\mathcal{G}_{n,p}$ using roughly the same number of colors. Therefore this type of analysis, though giving interesting results about random graphs, reveals nothing really useful about algorithms and their computational power. It is much more interesting to study another simple random graph model $\mathcal{G}_{n;k}$, which consists in picking up uniformly a random element of the class of all k-colorable graphs with n vertices. It was shown in [22] for $k < (1-\varepsilon)\log_2 n$, and generalized in [15] for $k = o(\sqrt{n/\log n})$, that this distribution can be approximated by the next procedure (using $p = 1/2$):

1. Generate a graph G using the distribution $\mathcal{G}_{n,p}$, (which means

that the probability that two vertices are connected by an edge is p, and these probabilites are independent for different pairs of vertices),

2. label vertices of G randomly by k colors,
3. remove all edges of G with endpoints labeled by the same label.

The graph distribution defined by this procedure is denoted by $\mathcal{G}_{n,p;k}$. Let us recall that we will suppose that p is a constant, but the results can be generalized (with some restrictions on p). Note that the first two phases of the construction are independent and hence the labeling of vertices can be performed first.

It is clear that the labeling constructed in the second step will become a coloring of G, and it is possible to prove that for $k = o(\sqrt{n/\log n})$ this labeling is almost surely both the optimal coloring and the unique (up to a permutation of colors) k-coloring of G.

No fast algorithm is known to give almost surely a good coloring of $\mathcal{G}_{n,p;k}$ for $k = \Omega(\sqrt{n})$. However the case of small k is quite appealing. Though primitive algorithms behave badly unless k is extremely small (e.g. the greedy algorithm uses almost surely $\Omega(n/\log n)$ colors to color $\mathcal{G}_{n,0.5;k}$ even if $k = n^{\varepsilon}$ for arbitrarily small positive constant ε [16]), more sophisticated algorithms are able to find the optimal solution almost surely (see [8] for k constant, [22] for $k < (1-\varepsilon)\log_2 n$, and [15] for $k = o(\sqrt{n/\log n})$).

The algorithm of [15] is based on dividing pairs of vertices into two classes using approximation of treshold functions with precision $1/k$. Such an approximation can be done in constant worst case time on CRCW parallel RAM, provided $k = \log^{O(1)} n$ [1, 12]. Based on this we will give a processor efficient CRCW PRAM algorithm coloring most k-colorable graphs optimally in constant time.

Since the treshold approximation with precision $1/k$ seems to be closely connected to evaluation of treshold functions, known lower bounds to depth of circuits computing symmetric functions [4, 19] suggest a conjecture that constant average time algorithms do not exist for coloring of k-colorable graphs if k grows faster than any polylogarithmic function.

We first present a constant time CRCW PRAM algorithm with a superpolynomial number of processors in Section 1. Section 2 shows how to find a coloring in constant time using small number of processors. However the algorithm is randomized (its main idea is to apply the algorithm of Section 1 to a small sample of $\log^{O(1)} n$ vertices and then to extent the solution to the whole graph). In Section 3 we show how to derandomize it to behave in the same way on random input graphs. This is based on the fact that the randomized algorithm does not ask for the existence of a large number of edges, and therefore this information can be used as a source of randomness.

1 Randomized algorithm with many processors

Let $0 < \vartheta < 1/2$ and $k = O(n^{\vartheta})$. In this paragraph we show an algorithm which finds almost surely a k-coloring of a random k-colorable graph with n vertices. The underlying idea of the algorithm comes from [15].

Given a vertex v, let us denote the class of equilabeled vertices obtained in the second step of the generating procedure $\mathcal{G}_{n,p;k}$ and containing v by C_v.

We will first show how, given a graph generated by the procedure $\mathcal{G}_{n,p;k}$ and its vertex v, we can find almost surely C_v:

```
**Algorithm $\mathcal{A}_0$.**
Input: A $k$-colorable graph $G$ with the vertex set $X$
        and the edge set $E$,
        a vertex $v \in X$.
Output: A color class $C$ of a $k$-coloring of $G$ containing $v$.
Remark: $\mathcal{A}_0$ sometimes gives an incorrect answer.
begin
for all $w \in X - \{v\}$ pardo
    $d(w) := |\{ z \in X \mid \{v,z\},\{w,z\} \in E \}|$;
for all $t := 1$ to $n$ pardo
    $\Gamma_t := \{v\} \cup \{ w \in X - \{v\} \mid d(w) \geq t \}$;
$t_v := \min\{ t \mid \Gamma_t$ is an independent set $\}$;
$C := \Gamma_{t_v}$;
end.
```

First we will prove some properties of $\mathcal{G}_{n,p;k}$.

Lemma 1.1 *Let $A \subset X$. With probability $1 - \exp(-\Omega(|A|/k))$, $\frac{|A|}{2k} < |C_v \cap A| < \frac{3|A|}{2k}$ for each vertex v.*

Proof: The size of the set $C_v \cap A$ is a random variable with binomial distribution $\mathcal{B}(|A|, \frac{1}{k})$. The Chernoff bound [5, 2] implies that

$$\mathbf{Prob}\left(\left| |C_v \cap A| - \frac{|A|}{k} \right| \geq \frac{|A|}{2k}\right) = \exp\left(-\Omega\left(\frac{|A|}{k}\right)\right).$$

♣

Corrolary 1.2 *With probability* $1 - \exp(-\Omega(n/k))$, $\frac{n}{2k} < |C_v| < \frac{3n}{2k}$ *for each vertex* v.

Lemma 1.3 *The probability that for each* v *the set* C_v *is a proper subset of no independent set of* $\mathcal{G}_{n,p;k}$ *is at least* $1 - nk(1-p)^{n/2k}$.

Proof: Given a fixed set of the form C_v and a fixed $x \notin C_v$, the probability that x has no neighbour in C_v is

$$(1-p)^{|C_v|} \leq (1-p)^{n/2k},$$

where we suppose $|C_v| \geq n/2k$, see the previous lemma. ♣

Let us now prove that the algorithm $\mathcal{A}_0$, applied to a vertex v of $\mathcal{G}_{n,p;k}$, returns almost surely C_v

Lemma 1.4 *If* $0 < \vartheta < 1/2$ *is a constant,* $k = O(n^\vartheta)$, *then the probability that* $\Gamma_\tau = C_v$, *where* $\tau = (n - |C_v| - \frac{n}{4k})p^2$, *is* $1 - \exp(-\Omega(n/k^2))$.

Proof: Let w be a vertex. Possible neighbours of both v and w are elements of $X - (C_v \cup C_w)$, each with probability p^2. Therefore it follows from the Chernoff inequality [5, 2] that

$$\left| d(w) - p^2 |X - (C_v \cup C_w)| \right| \geq \frac{n}{4k} p^2 \tag{1}$$

with probability at most

$$2\exp\left(-\frac{1}{2}\left(\frac{np^2}{4k}\right)^2 \frac{1}{n - |C_v \cup C_w|}\right) \leq 2\exp\left(-\frac{p^4}{32}\frac{n}{k^2}\right),$$

Suppose that (1) holds for all w, which happens with probability at least

$$1 - 2n\exp\left(-\frac{p^4}{32}\frac{n}{k^2}\right) = 1 - \exp\left(-\Omega\left(\frac{n}{k^2}\right)\right).$$

If $C_v = C_w$, then (1) implies

$$d(w) > (n - |C_v|)p^2 - \frac{n}{4k}p^2 = \tau,$$

if $C_v \neq C_w$, then

$$d(w) < (n - |C_v| - |C_w|)p^2 + \frac{n}{4k}p^2 \leq (n - |C_v| - \frac{n}{2k})p^2 + \frac{n}{4k}p^2 = \tau,$$

which implies $\Gamma_\tau = C_v$. ♣

Theorem 1.5 *The probability that the algorithm $\mathcal{A}_0$ returns C_v is* $1 - \exp(-\Omega(n/k^2))$.

Proof: Lemma 1.4 implies that Γ_t is likely to be C_v for some t. In view of Lemma 1.3, C_v is unlikely to be contained in a larger independent set, and therefore $\Gamma_t = C_v$. ♣

It is not surprising that the algorithm is likely to find the color class used in the construction $\mathcal{G}_{n,p;k}$, because it can be proved ([15, 22]) that the original coloring is almost surely the unique coloring of $\mathcal{G}_{n,p;k}$ (up to a permutation of colors).

The bounds to the resource requirements of the algorithm will be derived in the following paragraph. Let us just mention that computation of $d(v)$ in constant time requires superpolynomial number of processors. However the algorithms of the next paragraphs will call $\mathcal{A}_0$ to find an independent set of a graph of a polylogarithmic size.

$\mathcal{A}_0$ gives immediately an algorithm to color $\mathcal{G}_{n,p;k}$ almost surely by k colors:

```
Algorithm $\mathcal{A}_1$.
Input: A $k$-colorable graph $G$ with the vertex set $X$
        ordered by a relation $<$ and the edge set $E$.
Output: A $k$-coloring of $G$.
Remark: $\mathcal{A}_1$ sometimes fails without giving an answer.
begin
for all vertices $v$ pardo begin
    find the set $C_v$ using $\mathcal{A}_0$;
    if $v = \min C_v$ then label $v$ as "selected"
    else label $v$ as "unselected";
    end;
```

```
for all selected vertices $v$ pardo begin
    $c_v :=$ the number of selected vertices $w$ such that $w < v$;
    color all vertices of $C_v$ by $c_v$;
    end;
for all edges $e = \{v, w\} \in E$ pardo
    if $v$ and $w$ are colored by the same color then
      the computation failed;
if there exists a selected vertex $v$ such that $c_v \geq k$ then
    the computation failed;
end.
```

Note that the algorithm $\mathcal{A}_1$ either reports a failure or gives a valid k-coloring of the graph by colors $0, 1, \ldots, k-1$. It is necessary to check the correctness of the result of the computation, because the algorithm $\mathcal{A}_0$ might give an incorrect answer.

It follows from the analysis of $\mathcal{A}_0$ that

Theorem 1.6 *The algorithm $\mathcal{A}_1$ finds a k-coloring of the input graph with probability* $1 - \exp(-\Omega(n/k^2))$.

Proof: With probability $1 - n\exp(-\Omega(n/k^2)) = 1 - \exp(-\Omega(n/k^2))$, all sets C_v are computed correctly by $\mathcal{A}_0$. ♣

2 Efficient randomized algoritm

In this paragraph we suppose that $k = \log^{O(1)}$. Let m be a number sufficiently larger than k^2. It will be quite sufficient to put $m = k^2 \log^a n$ for some constant $a \geq 2$. For reasons that will be clear in the next paragraph, we also choose a fixed set Q of, say, $n/2$ vertices X.

We will show that a solution of the coloring problem for a random subset of m vertices of $X - Q$ gives almost surely a solution of the global problem.

Graph coloring algorithm $\mathcal{A}_2$.

```
Input: A $k$-colorable graph $G$ with vertex set $X$ of size $n$
       and edge set $E$, $k = \log^{O(1)} n$,
       random bits $r_{i,j}$, $i = 1, \ldots, s$, $j = 1, \ldots, \lceil \log_2 n \rceil$
       a set $Q$ of $\lfloor n/2 \rfloor$ vertices of $G$,
    a constant $a$.
Output: A coloring of $G$.
Remark: $\mathcal{A}_2$ sometimes fails to find a $k$-coloring.
begin
choose a number $m$ such that $m \geq k^2 \log^a n$;
for $i = 1$ to $m$ pardo
    $w_i :=$ the number with binary representation $r_{i,1} \ldots r_{i,t}$,
    where $t = \lceil \log_2 n \rceil$;
$Y := \{w_i \mid 1 \leq i \leq m,\ w_i \in X - Q\}$;
use $\mathcal{A}_1$ to find a $k$ coloring of the graph induced by $G$ on $Y$;
if $\mathcal{A}_1$ fails then halt;
for each vertex $v$ choose an integer $0 \leq \lambda(v) < k$ such that
    there is no $w \in Y$ such that
    $\{v, w\}$ is an edge of $G$ and $w$ is colored by $\lambda(v)$;
if such $\lambda(v)$ does not exit for some vertex $v$ then halt;
if $\lambda$ is not a coloring of $G$ then halt;
color each vertex $v$ of $G$ by $\lambda(v)$;
end.
```

Note that if $v, w \in Q$, the algorithm never asks whether v and w are connected by an edge of G.

Randomized algorithms are usually analyzed on assumption that a source of randomness provides independent random bits such that $\mathbf{Prob}(r_i = 1) = 1/2$ for each i. The random bits constructed in the next paragraph are independent, but it could only be proved that the probability of being equal to 1 is closed to $1/2$.

We first prove that the size of any set $C_v \cap Y$ is sufficiently large:

Lemma 2.1 *Let us suppose that random bits $r_{i,j}$, given as an input to $\mathcal{A}_2$, are independent and $\mathbf{Prob}(r_{i,j} = 1) = (1 + O(\log^{-1} n))/2$ for all i, j. There exists a constant $c > 0$ such that, with probability $1 - \exp(-\Omega(m/k))$, $|C_v \cap Y| \geq cm/k$ for each vertex v.*

Proof: Let $\kappa = O(\log^{-1} n)$ be such that $\mathbf{Prob}(r_{i,j} = 1), \mathbf{Prob}(r_{i,j} = 0) \geq (1 - \kappa)/2$ for all i, j. There is a constant $\gamma > 0$ such that

$$\mathbf{Prob}(w_j = z) \geq \left(\frac{1-\kappa}{2}\right)^{\lceil \log_2 n \rceil} \geq \frac{\gamma}{n}$$

for each vertex z and $j \leq m$.

Let v be a fixed vertex of $X - Q$. It follows from Lemma 1.1 that we can suppose that $|C_v - Q| \geq |X - Q|/2k \geq n/4k$, and therefore

$$\mathbf{Prob}(w_j \in (C_v - Q - \{w_\ell | \ell < j\})) \geq \left(\frac{n}{4k} - m\right)\frac{\gamma}{n} \geq \frac{\gamma}{5k}.$$

In view of the Chernoff bound

$$\mathbf{Prob}\left(|C_v \cap Y| < \frac{\gamma m}{6k}\right) \leq \exp\left(-\Omega\left(\frac{m}{k}\right)\right),$$

and the probability that this is true for all k color classes is not more that k times greater. ♣

Note that the Lemma implies that $|Y - Q| \geq cm$ with large probability.

Theorem 2.2 *Under the same assumptions on bits $r_{i,j}$ as in the preceeding Lemma, if $k = \log^{O(1)} n$, then the probability that the algorithm $\mathcal{A}_2$ finds a k-coloring of $\mathcal{G}_{n,p;k}$ is at least* $1 - \exp(-\Omega(m/k^2)) = 1 - \exp(-\Omega(\log^a n))$.

Proof: Since the algorithm constructs the set Y before it asks for any edge, Y can be determined before the construction $\mathcal{G}_{n,p;k}$ is applied. It follows that the probabilistic distribution of the graph induced on Y is the same as $\mathcal{G}_{m,p;k}$, and therefore $\mathcal{A}_1$ finds the solution given by classes $C_1 \cap Y, \ldots, C_k \cap Y$ with probability at least

$$1 - \exp(-\Omega(m/k^2)) \geq 1 - \exp(-\Omega(\log^a n),$$

see Theorem 1.6. If this is the case, the choice of $\lambda(v)$ is possible for each v as the color used for $C_v \cap Y$. Now, it is sufficient to prove that no other choice of $\lambda(v)$ is likely for any vertex v. In view of the preceeding lemma, the probability that a vertex v is connected to no $w \in C_z \cap Y$, $v \notin C_z$, is at most

$$(1-p)^{cm/k} = \exp(-\Omega(\log^a n).$$

♣

We are interested in three resources used by the algorithm $\mathcal{A}_0$: the worst case parallel time, the number of processors and the amount of randomness. We will need the following technical lemma, that says that it is possible to add a polylogarithmic number of bits in constant time with small number of processors. It is known that there are families of constant depth circuits that compute the sum of n bits, provided the number of 1's among them is bounded by a polylogarithmic function [1, 12]. Unfortunately the proofs are either nonconstructive ([1]), or possess a certain degree of uniformness, but they are complicated. However in our special case there is an easy direct proof, since the most difficult part of the general proof is a compaction of 1's into an array of a polylogarithmic size, which is already done here.

Lemma 2.3 *Let $0 < \varepsilon, \ell$ be constants. If $i \leq \lfloor \log^{\ell} n \rfloor$, then the sum of i integers $b_1, \ldots, b_i \in \{0, 1\}$ can be computed in constant time on CRCW PRAM with $O(n^{\varepsilon})$ processors.*

Proof: In constant time and using $O(AC^{-1}B^{C})$ processors, we can reduce computing of the sum of A integers $0 \leq b_1, \ldots, b_A < B$ to computing the sum of at most A/C integers less or equal to BC:

Partition numbers into A/C groups of at most C elements each. There are at most B^C different sequences σ of C numbers from the interval $[0, B)$. For each of A/C groups of numbers, use B^C families of processors, such that for each sequence σ there is exactly one family that checks whether the numbers in the group form a sequence σ and, in the affirmative case, outputs the predefined result.

Now put $C = \log^{1/2} n$. If $B \leq A = \log^{O(1)} n$, then

$$AC^{-1}B^{C} \leq A^{C+1} = \exp((C+1)\log A) = \exp(O(\log^{1/2} n \log\log n)) =$$

$$= O(\exp(\varepsilon \log n)) = O(n^{\varepsilon}).$$

Repeating the reduction 2ℓ times, we can compute the sum of $\log^{\ell} n$ nembers from $\{0, 1\}$ in constant time with $O(n^{\varepsilon})$ processors. ♣

Theorem 2.4 *Let $k = \log^{O(1)} n$. There is an implementation of $\mathcal{A}_2$ on CRCW PRAM with $O(|V| + |E|)$ processors that runs in constant parallel time and needs $\log^{O(1)} n$ random bits.*

Proof: $\mathcal{A}_2$ calls $\mathcal{A}_1$ (and through it $\mathcal{A}_0$) on a subgraph of polylogarithmic size. It follows from the preceeding lemma that, given a constant $\varepsilon > 0$, all sums computed by $\mathcal{A}_0$ and $\mathcal{A}_1$ in such a case can be computed in constant parallel time using $O(n^{\varepsilon})$ processors.

It is known that the minimum of m bits can be computed in constant parallel time on CRCW PRAM with $O(m^2)$ processors [15].

The algorithm $\mathcal{A}_0$ checks if $d(w) \geq t$ for each t, w. This can be done using n^2 computations based on the preceeding Lemma. Remaining part of the computation can easily be done in constant time on CRCW PRAM with polynomially many processors. ♣

3 Derandomization

The algorithms given in the previous two paragraphs are randomized. We will now show a way of derandomizing the coloring algorithm $\mathcal{A}_2$.

Choose a fixed vertex $u \in Q$ (for the set Q, see Algorithm $\mathcal{A}_2$). Given a vertex $v \in Q - \{u\}$, put $b_v = 1$ if $\{u, v\}$ is an edge, $b_v = 0$ otherwise. The key observation is that $\mathcal{A}_2$ checks the value of b_v for no $v \in Q - \{u\}$. If the input graph is generated by $\mathcal{G}_{n,p;k}$, we can use boolean variables b_v, $v \in Q - \{u\}$ as a source of random bits. The probabilities $\mathbf{Prob}(b_v = 1)$ are independent, but the problem is that they are not equal to $1/2$.

Therefore we will group bits b_v into groups of polylogarithmic size, and we will use the parities of groups instead of the single bits. The parity of $\log^{O(1)} n$ bits can be computed in constant time using $O(n^\epsilon)$ processors, see Lemma 2.3, bits obtained in this way are independent, and they are almost unbiased:

Lemma 3.1 *Let $v_1, \ldots, v_\ell$ are different elements of $Q - \{u\}$. Then*

$$\mathbf{Prob}(b_{v_1} \oplus \cdots \oplus b_{v_\ell} = 1) = \frac{1}{2}(1 + \exp(-\Omega(\ell/k))).$$

Proof: The probability that $b_v = 1$ is either 0 if $v \in C_u$ or p if $v \notin C_u$. In view of Lemma 1.1, the probability that C_u contains more than $3\ell/2k \leq \ell/2$ vertices v_i, $1 \leq i \leq \ell$ is $\exp(-\Omega(\ell/k)$.

Suppose now that $w_1, \ldots, w_{\ell/2}$ is a subsequence of $v_1, \ldots, v_\ell$ that does not contain points of C_u. Then

$$\mathbf{Prob}(b_{w_i} = 1) = \frac{1}{2}(1 + (2p - 1)),$$

$$\mathbf{Prob}(b_{w_1} \oplus \cdots \oplus b_{w_\ell/2} = 1) = \frac{1}{2}(1 + (2p - 1)^{\ell/2}),$$

and therefore given any fixed values of b_{v_j} for v_j that do not belong to the sequence w_i,

$$\mathbf{Prob}(b_{v_1} \oplus \cdots \oplus b_{v_\ell} = 1) = \frac{1}{2}(1 \pm (2p - 1)^{\ell/2}) = \frac{1}{2}(1 + \exp(-\Omega(\ell))).$$

♣

Theorem 2.4 and the preceeding lemma give

Theorem 3.2 *Let $k = \log^{O(1)} n$, a be a constant. There is a deterministic CRCW PRAM algorithm that runs in constant parallel time and, with probability $1 - n^{\Omega(\log^a n)}$, colors the random graph $\mathcal{G}_{n,p;k}$ with k colors.*

Proof: Choose $\ell = k \log^{a+2} n$. It follows from Lemma 3.1 that, with probability at least

$$1 - (\log^{O(1)} n) \exp(-\Omega(\ell/k)) = 1 - n^{\Omega(\log^a n)},$$

a fixed polylogarithmic number of groups of ℓ bits b_v give a sequence of random bits that verifies assumption of Theorem 2.4. ♣

4 Conclusions

We have proved that $\mathcal{G}_{n,p;k}$, $k = \log^{O(1)} n$, can be colored almost surely with k colors using a constant time CRCW PRAM with $O(|V|+|E|) = O(n^2)$ processors. This means that both time and processor bounds are optimal. Using methods of [8], it is easy to obtain a CRCW PRAM algorithm, which colors $\mathcal{G}_{n,p;k}$ *always* with k colors, such that its *average* running time is constant, and the number of processors equal to the number of vertices and edges of the graphs (it is sufficient to use an $n^{polylogn}$ expected time algorithm to color the input graph when $\mathcal{A}_2$ fails). In view of [22, 15], the same results are true for a graph drawn randomly (with uniform probability) from the class of all k-colorable graphs with n vertices, $k = \log^{O(1)} n$.

Our result are closely connected to the existence of uniform constant - depth probabilistic circuits, approximating the treshold functions. If treshold functions with n inputs can be approximated with multiplicative precision $1/k$, where $k = o(\sqrt{n/\log n})$, then k-colorable graphs can be colored almost surely with k colors in constant average time, see Lemma 1.4.

Several problems are left open

- Is it possible to find a constant average time algorithm coloring $\mathcal{G}_{n,p;k}$ by k colors if k is not bounded by a polylogarithmic function?

- What is the best precision bound to constant-depth probabilistic circuits, approximating general treshold functions, and what is the optimal size of such circuits ?

The results show that methods based on taking small random samples of the input information can give very efficient probabilistic approximation algorithms and deterministic algorithms with good behavior on randomly generated inputs. It is likely that this paradigm can be used to solve other problems.

References

[1] M. Ajtai, and M. Ben-Or, A theorem on probabilistic constant depth computation, *Proceedings of the 16th Symposium on Theory of Computing*, (1984), 471-474.

[2] N. Alon, J. Spencer, and P. Erdős, The probabilistic Method, J. Wiley and Sons, New York, 1992.

[3] B. Bollobás, The chromatic number of random graphs, *Combinatorica* **8**, 49-56.

[4] B. Brustmann, and I. Wegener, The complexity of symmetric functions in bounded depth circuits, *Information Processing Letters* **25** (1987), 217-219.

[5] H. Chernoff, A measure of asymptotic efficiency for tests based on the sum of observations, *Ann. Math. Statist.* **23** (1952), 493-509.

[6] B. Chlebus, K. Diks, T. Hagerup, and T. Radzik, Efficient simulations between CRCW PRAMs, *Proc. 13th Symp. on the Mathematical Foundations of Computer Science*, 1988, 230-239.

[7] D. Coppersmith, P. Raghavan, and M. Tompa, Parallel graph algorithms that are efficient on average, *Proceedings of the 28th Annual IEEE Conference on Foundations of Computer Science*, (1987), 260-269.

[8] M. Dyer, and A. Frieze, The solution of some random NP-hard problems in polynomial expected time, *J.Algorithms* **10** (1989), 451-489.

[9] A. Frieze, and L. Kučera, Parallel colouring of random graphs, in *Random Graphs* **87**, M. Karonski, J. Jaworski, A. Rucinski, eds, J. Wiley 1990, 41-52