Random Graphs '87
Edited by M. Karoński, J. Jaworski and A. Ruciński

PARALLEL COLOURING OF RANDOM GRAPHS

Alan M. FRIEZE

Department of Mathematics, Carnegie–Mellon University, Pittsburgh, USA

and

Luděk KUČERA

Charles University, Prague, Czechoslovakia

The paper describes a parallel implementation of the greedy colouring algorithm which colours almost all graphs with n vertices by at most

$$\left(1+\frac{1}{\log_2 \log_2 \log_2 n}\right)\frac{n}{\log_2 n}$$

colours (the probability that a random graph with n vertices is coloured by more colours is

$$o(n^{-\log_2 \log_2 n/(3 \log_2 \log_2 \log_2 n)}) = o(n^{-k})$$

for each fixed k). The algorithm runs in $O(\log_2^2 n \log_2 \log_2 n)$ worst case parallel time on PRIORITY concurrent-read concurrent-write parallel RAM with $O(n^2 \log_2^{-2} n)$ processors and consequently in $O(\log_2^3 n \log_2 \log_2 n)$ worst case parallel time on exclusively-read exclusively-write parallel RAM. The expected number of colours used by the algorithm is roughly the same as the number of colours used by the sequential greedy algorithm and approximately twice the expected value of the chromatic number of the graph.

1. Introduction

Colouring of graphs is known to be NP-complete [Ka], [GJ] and therefore no algorithm is known to find the optimal colouring of graphs in polynomial worst-case time. Moreover, it has been proved that the existence of a polynomial time algorithm colouring any graph G by at most $1.99\ \chi(G)$ colours would imply P=NP. However, there are fast graph

colouring algorithms with a good average performance. Perhaps the simplest one among them is the greedy colouring. It was proved that the average value of the ratio $A(G)/\chi(G)$, where $A(G)$ denotes the number of colours used by the algorithm, is $2+o(1)$ (for the upper bound see [GMcD]). This means that, in spite of its simplicity and speed, the greedy colouring is a relatively good algorithm. Moreover, no polynomial time algorithm is known to have substantially better average behaviour on the class of all graphs.

The aim of the present paper is to study a possibility to implement the greedy colouring algorithm in poly-log parallel time. A colouring can be viewed as a covering of the vertex set by disjoint stable sets, because monochromatic sets of vertices are stable. Since the expected size of the largest stable set of almost all graphs with n vertices is $(2+o(1))\log_2 n$, it is easy to implement the greedy algorithm for finding large stable sets in poly-log parallel time. However, the greedy *colouring* algorithm seems to be inherently sequential, because the condition of disjointness implies that stable sets can not be looked for simultaneously.

Our algorithm is based on the idea to cover the vertex set by stable sets which are not necessarily disjoint, colour each stable set by one colour (of course, some vertices are coloured by more than one colour) and finally to choose arbitrarily one of the assigned colours for each vertex. The quality of such an algorithm depends strongly on the amount of vertices which are coloured by more than one colour.

The basic result of the paper, Lemma 1, shows that, roughly speaking, if we construct independently and greedily about $n/\log^2 n$ maximal stable sets in a suitable way then their overlapping is almost surely sufficiently small and the number of covered vertices is of order $n/\log n$. Now, it is not difficult to see that poly-log number of iterations of this procedure gives a good colouring of all vertices of the input graph.

Throughout the paper, $0<p<1$ is a constant and $\log n$ means $\log_b n$, where $b=(1-p)^{-1}$, V_n denotes the set $\{1,\ldots,n\}$.

As a model of random graphs we use the probability space $G_{n,p}$ of all labelled graphs with the vertex set V_n, where

$$\mathbf{Prob}(G_{n,p}=G)=p^m(1-p)^{\binom{n}{2}-m}$$

for each graph G with the vertex set V_n and m edges. In other words, the probability that two vertices are connected is p and those probabilities are independent for different pairs of vertices.

As a model of computing we use a concurrent-read concurrent-write parallel random access machine with PRIORITY rule of writing conflict resolution [Ku] and unit cost of operations. More precisely, we suppose that each cell of memory of parallel RAM can contain a natural number of size $O(n)$, where n is the number of vertices of the input graph, the cost of any arithmetic operation is constant, the number of processors is bounded by a polynomial in n, any number of processors can read simultaneously from the same memory cell and, moreover, processors are linearly ordered as $P_1,...,P_k$ and if several processors try to write into the same memory cell then the processor with the smallest order number succeeds.

2. A Parallel Greedy Colouring Algorithm

In this section we describe our parallel greedy colouring algorithm. Our algorithm proceeds in rounds. In the first round $L=\lceil \log n \rceil^2$ and $m=\lfloor n/L \rfloor$ and we have m (groups of) processors $P_1,...,P_m$ which independently and greedily choose stable sets $S_1, S_2,..., S_m$ where processor P_i starts its construction at $k_i=(i-1)L+1$ and examines vertices in order $k_i, k_i+1, k_i+2,...,n, 1,..., k_i-1$.

We colour $v \in S=S_1 \cup ... \cup S_m$ with colour t_v where t_v is an arbitrary element of $\{i | v \in S_i\}$, and repeat the process on $G[V_n-S]$. The main task is to show that the size of S is approximately $m \log n$ with high probability. It then follows easily that $G_{n,p}$ can usually be coloured with approximately $n/\log n$ colours in $O(\log n)$ rounds.

The algorithm is now given more formally.

Parallel Greedy Colouring Algorithm (PGC)

1. **begin**
2. $V_R := V_n$ {=set of uncoloured vertices};
3. $c_0 := 0$;
4. **for** $j := 1$ **to** n **pardo** $c(j) := 0$;
5. {$c(j)$ is to become the colour of vertex j}
6. **while** $|V_R| \geqslant \frac{n}{\log n \log\log n}$ **do**
7. **begin**
8. $L := \lceil \log |V_R| \rceil^2$;
9. $m := \lfloor |V_R|/L \rfloor$;

10. **for** $i:=1$ **to** m **pardo** GREEDY (c_0, i);
11. $S:=\{v|c(v)>c_0\}$; $V_R:=V_R-S$; $c_0:=c_0+m$;
12. **end;**
13. {assume $V_R=\{v_1,v_2,\ldots,v_r\}$}
14. **for** $s:=1$ **to** r **pardo** $c(v_s):=c_0+s$;
15. **end;**

where procedure GREEDY(c,i), which chooses greedily a stable set contained in $V_R=\{v_1,v_2,\ldots,v_r\}$ starting at vertex $(i-1)L+1$ and colours its elements by $c+i$ is defined as follows:

procedure GREEDY(c,i);

1. **begin**
2. $k_i:=(i-1)L+1$; $X:=\{v_{k_i}\}$;
3. **while** (X is not a maximal stable set in $G[V_R]$) and ($|X|<3\log n$) **do**
4. **begin**
5. $l:=\min\{k|v_{k_i+k \bmod r}\in X$ and $\{v_{k_i+k \bmod r}\}\cup X$ is stable$\}$;
6. $X:=X\cup\{v_{k_i+l \bmod r}\}$;
7. $c(v_{k_i+l \bmod r}):=c+i$;
8. **end;**
9. **end;**

Note that several processors (corresponding to simultaneously executed calls of GREEDY(c,i) for different i) might try to write simultaneously into the same variable (i.e. to colour the same vertex) in line 7 of GREEDY; this is of course allowed on PRIORITY concurrent-read concurrent-write parallel RAM. Note that RANDOM write conflict resolution rule (a randomly chosen processor succeeds if several of them try to write into the same memory cell) would be sufficient to implement colouring in line 7 of GREEDY, but the PRIORITY rule is very convenient to implement line 5 of GREEDY, as we shall see later.

3. Analysis of Parallel Greedy Colouring

The following lemma is the key to the analysis. Its proof is deferred until we have explored its consequences.

Lemma 1. *Let m,S refer to the* first *execution of the* while *loop* 7-12. *Then*

$$\mathbf{Prob}(|S| \leqslant (1-\frac{1}{\log\log n})m\log n) = O\left(\frac{(\log\log n)^3}{\log n}\right).$$

Armed with Lemma 1 we can prove

Theorem 1. *If* PGC *is applied to* $G_{n,p}$ *then*

(a) $\mathbf{Prob}$[PGC uses more than $\left(1+\frac{1}{\log\log\log n}\right)\frac{n}{\log n}$ colours]
$= o(n^{-\frac{1}{3}\frac{\log\log n}{\log\log\log n}})$.

(b) $\mathbf{Prob}$[the number of iteration of while loop 7-12 is at least $\left(1+\frac{3\log\log\log n}{\log\log n}\right)\log n\log\log n$]
$= o(n^{-\frac{1}{3}\frac{\log\log n}{\log\log\log n}})$.

Proof. We rely on the fact that an iteration of while loop 7-12 does not condition the edges contained in V_R. This is a well known property of the greedy algorithm for constructing a stable set.

Let n_t denote $|V_R|$ at the start of the t'th iteration of the while loop. Thus $n_1 = n$ and

$$n_t \geqslant \frac{n}{\log n\log\log n}$$

if the loop is executed at least t times.

Let $\delta_t = 1$ if

$$n_t - n_{t+1} < (1-\frac{1}{\log\log n_t})m_t\log n_t,$$

where $m_t = \lfloor n_t/L_t \rfloor$ and $L_t = \lceil \log n_t \rceil^2$, otherwise $\delta_t = 0$. Suppose that the while loop is executed T times. Then the number of colours K used satisfies

$$K \leqslant \sum_{t=1}^{T} m_t + \frac{n}{\log n\log\log n} \leqslant \sum_{t=1}^{T}(1-\delta_t)m_t + n_1\sum_{t=1}^{T}\delta_t + \frac{n}{\log n\log\log n}. \tag{1}$$

But since

$$n_t - n_{t+1} \geqslant (1-\delta_t)\left(1-\frac{1}{\log\log n_t}\right) m_t \log n_t$$

$$\geqslant (1-\delta_t)\left(1-\frac{2}{\log\log n}\right) m_t \log n \quad \text{for } t < T \text{ and } n \text{ large},$$

we have

$$\sum_{t=1}^{T} (1-\delta_t) m_t \leqslant \left(1+\frac{3}{\log\log n}\right)\frac{n}{\log n} \quad \text{for } n \text{ large}. \tag{2}$$

Now if n is large then $\delta_t = 0$ implies

$$n_{t+1} \leqslant \left(1-\frac{1-\frac{2}{\log\log n}}{\log n}\right) n_t .$$

Hence if $T' = \sum_{t=1}^{T} (1-\delta_t)$ then we have

$$\frac{n}{\log n \log\log n} \leqslant n\left(1-\frac{1-\frac{2}{\log\log n}}{\log n}\right)^{T'} \leqslant n e^{-T'(1-2/\log\log n)/\log n}.$$

For large n this implies

$$T' \leqslant \log n \log\log n \left(1+\frac{2\log\log\log n}{\log\log n}\right). \tag{3}$$

Now $\delta_1, \delta_2, \ldots, \delta_T$ is a sequence of independent Bernoulli trials where

$$\mathbf{Prob}(\delta_T = 1) \leqslant \frac{a(\log\log n)^3}{\log n}$$

for some constant $a > 0$. (Note that $\log n_t \approx \log n$ is used here.)

Now (3) implies that if

$$T \geqslant T_0 = \left\lceil \log n \log\log n \left(1+\frac{3\log\log\log n}{\log\log n}\right)\right\rceil$$

that $\delta_t=1$ at least $\log n \log\log n \gg t_0 = \lceil \frac{\log n}{2\log\log\log n} \rceil$ times in the first T_0 trials. The probability of t_0 successes in T_0 trials is at most

$$\binom{T_0}{t_0}\left(\frac{a(\log\log n)^3}{\log n}\right)^{t_0} \leqslant \left(\frac{T_0 e a(\log\log n)^3}{t_0 \log n}\right)^{t_0} = o(n^{-\frac{1}{3}\frac{\log\log n}{\log\log\log n}}).$$

This proves (a) and then (b) follows from (1), (2) and the above. □

The following lemma establishes some (possibly well known) facts about the (ordinary) greedy algorithm.

Lemma 2. *Let $\varepsilon>0$ be arbitrary.*

$$\mathbf{Prob}(|S_1| \leqslant (1-\varepsilon)\log n) \leqslant n e^{-n^{\varepsilon}/\log n} \tag{3a}$$

$$\mathbf{Prob}(|S_1| \geqslant (1+\varepsilon)\log n) \leqslant n^{-\varepsilon}/q, \quad \textit{where } q=1-p, \tag{3b}$$

$$\mathbf{E}(|S_1|) = (1+\theta_n)\log n, \quad \textit{where } |\theta_n| = O(\log\log n/\log n), \tag{3c}$$

$$\mathbf{Prob}(k \in S_1) \leqslant \frac{a\log\log k}{k} \quad \textit{for } 2 \leqslant k \leqslant n, \tag{3d}$$

where $a>0$ is a constant.

Proof. (a) Let E_i denote the event

$$\{i, i+1, \ldots, i+\tfrac{n}{\lfloor(1-\varepsilon)\log n\rfloor}-1 \text{ are not adjacent to } S_1 \cap \{1,2,\ldots,i-1\}$$
$$\text{and } |S_1 \cap \{1,2,\ldots,i-1\}| \leqslant (1-\varepsilon)\log n\}$$
$$\text{for } 1 \leqslant i \leqslant n_0 = n+1-\tfrac{n}{\lfloor(1-\varepsilon)\log n\rfloor}.$$

Now the event

$$\{|S_1| \leqslant (1-\varepsilon)\log n\} \subset \bigcup_{i=1}^{n_0} E_i$$

and

$$\mathbf{Prob}(E_i \mid S_1 \cap \{1,2,\ldots,i-1\}) \leqslant (1-(1-p)^{(1-\varepsilon)\log n})^{\frac{n}{(1-\varepsilon)\log n}}$$
$$= (1-n^{-(1-\varepsilon)})^{\frac{n}{(1-\varepsilon)\log n}} \leqslant e^{-n^{\varepsilon}/\log n}.$$

Now (a) follows on removing the conditioning and multiplying by n_0.

(b) Let F_i denote the event

$$\{i \text{ is not adjacent to } S_1 \cap \{1,2,\ldots,i-1\} \text{ and } |S_1 \cap \{1,2,\ldots,i-1\}| \geqslant \lfloor (1+\varepsilon)\log n \rfloor - 1\}.$$

Then

$$\mathbf{Prob}(|S_1| \geqslant (1+\varepsilon)\log n) \leqslant \mathbf{Prob}\left(\bigcup_{i=1}^{n} F_i\right) \leqslant n(1-p)^{(1+\varepsilon)\log n - 1} = n^{-\varepsilon}/q.$$

(c) Put

$$\varepsilon = \varepsilon_a = \frac{\log_e 2 + \log_e \log_b n + \log_e \log_e n}{\log_e n}.$$

Then $n^{\varepsilon} = 2(\log_b n)(\log_e n)$ and so

$$\mathbf{Prob}(|S_1| \leqslant (1-\varepsilon_a)\log n) \leqslant \frac{1}{n}.$$

Hence

$$\mathbf{E}(|S_1|) \geqslant (1-n^{-1})(1-\varepsilon_a)\log n. \tag{4}$$

Now put $\varepsilon = \varepsilon_b = \log_e \log_b n / \log_e n$ in (b) so that $n^{\varepsilon} = \log_b n$. Then

$$\begin{aligned}\mathbf{E}(|S_1|) &\leqslant (1+\varepsilon_b)\log n + 3\log n\, \mathbf{Prob}((1+\varepsilon_b)\log n < |S_1| \leqslant 3\log n) \\ &\quad + n\, \mathbf{Prob}(|S_1| > 3\log n) \qquad (5) \\ &\leqslant (1+\varepsilon_b)\log n + 3/q + n^{-1}/q,\end{aligned}$$

as (b) implies $\mathbf{Prob}(|S_1| \geqslant 3\log n) \leqslant n^{-2}/q$. (c) follows from (4) and (5).

(d) Let $\pi_k = \mathbf{Prob}(k \in S_1)$ for $1 \leqslant k \leqslant n$. Then

$$\mathbf{E}(|S_1|) = \pi_1 + \pi_2 + \ldots + \pi_n. \tag{6}$$

Furthermore

$$\pi_k \geqslant \pi_{k+1} \quad \text{for } 1 \leqslant k < n \tag{7}$$

since

$$\mathbf{Prob}(k+1 \in S_1) = \mathbf{Prob}(k+1 \text{ is not adjacent to } S_1 \cap \{1,2,\ldots,k\}) \leqslant$$

$$\leqslant \mathbf{Prob}(k+1 \text{ is not adjacent to } S_1 \cap \{1,2,\ldots,k-1\})$$
$$= \mathbf{Prob}(k \in S_1)\,.$$

(6) and (c) yield

$$\pi_1+\pi_2+\ldots+\pi_{\lfloor n/2\rfloor} \leqslant (1-\theta_{\lfloor n/2\rfloor})\log\lceil n/2\rceil\,,$$
$$\pi_1+\pi_2+\ldots+\pi_n \leqslant (1+\theta_n)\log n$$

and hence

$$\pi_{\lfloor n/2\rfloor+1}+\pi_{\lfloor n/2\rfloor+2}+\ldots+\pi_n \leqslant \log\frac{n}{\lceil n/2\rceil}+\theta_n\log n+\theta_{\lfloor n/2\rfloor}\log\lceil n/2\rceil$$
$$\leqslant A\log\log n$$

for some $A>0$. (7) now implies

$$\pi_n \leqslant \frac{A\log\log n}{n-\lceil n/2\rceil}$$

and (d) follows. □

We can now turn to the

Proof of Lemma 1. We use the fact that

$$|S| \geqslant \sum_{i=1}^{m}|S_i| - \sum_{i=1}^{m-1}\sum_{j=i+1}^{m}|S_i\cap S_j|. \tag{8}$$

It follows from (3a) with $\varepsilon=(\log n)^{-1/2}$ that

$$\mathbf{Prob}\Big(\sum_{i=1}^{m}|S_i| \leqslant (1-(\log n)^{-1/2})m\log n\Big)$$
$$\leqslant mn\exp\{-n^{-(\log n)^{-1/2}}/\log n\} = o(n^{-\log n}). \tag{9}$$

Next let $R(i,j)=1$ if $j\in S_i$, $R(i,j)=0$ otherwise, for $1\leqslant i\leqslant m$, $1\leqslant j\leqslant n$. Then

$$\sum_{i=1}^{m-1}\sum_{j=i+1}^{m}|S_i\cap S_j| = \sum_{k=1}^{n}W_k, \tag{10}$$

where

$$W_k=\sum_{i=1}^{m-1}\sum_{j=i+1}^{m} R(i,k)R(j,k)\,.$$

Let $\psi(k)=(k-\max\{k_i|k_i\leqslant k\}) \bmod n+1$. We show later that

$$\mathrm{E}(|W_k|)=\mathrm{o}\left(\frac{(\log\log n)^2}{L}+\frac{(\log\log n)^2}{\psi_k\log n}\right). \tag{11}$$

This implies that

$$\mathrm{E}\left(\sum_{k=1}^{n}|W_k|\right)=\mathrm{O}\left(\frac{n(\log\log n)^2}{(\log n)^2}\right)+\mathrm{O}\left(\frac{(\log\log n)^2}{\log n}\right)\sum_{k=1}^{n}\frac{1}{\psi(k)}$$

$$=\mathrm{O}(m(\log\log n)^2)\,,$$

since $\sum_{k=1}^{n}1/\psi(k)=\mathrm{O}(m\log\log n)$. Hence, the Markov inequality implies that

$$\mathbf{Prob}\left(\sum_{k=1}^{n}|W_k|\geqslant\frac{m\log n}{2\log\log n}\right)=\mathrm{O}\left(\frac{(\log\log n)^3}{\log n}\right). \tag{12}$$

(8), (9), (10) and (12) together imply the lemma. We must now prove (11).

Consider the sequence of vertices $k_i, k_i+1,\ldots,n,1,\ldots,k_i-1$ looked at by processor P_i. Let $\alpha(i,k)$ denote the position of k in this sequence so that $\alpha(i,k_i)=1$ etc. Now,

$$\mathrm{E}(R(i,k)R(j,k))=\mathbf{Prob}(k\in S_j)\,\mathbf{Prob}(k\in S_i|k\in S_j). \tag{13}$$

Assume that $\alpha(i,k)>\alpha(j,k)$. Then, by (3d),

$$\mathbf{Prob}(k\in S_j)\leqslant\frac{a\log\log n}{\alpha(j,k)} \tag{14}$$

and

$$\mathbf{Prob}(k\in S_i|k\in S_j)$$

$$\leqslant\mathbf{Prob}(k\text{ is not adjacent to }S_i\cap\{k_i,k_i+1,\ldots,k_j-1\})\leqslant$$

$$\leqslant \frac{a \log\log n}{\alpha(i,k)-\alpha(j,k)} \leqslant \frac{a \log\log n}{((j-i) \bmod m)L}. \tag{15}$$

We can assume w.l.o.g. that $k \geqslant k_m$ and then (13)-(15) imply

$$\begin{aligned} \mathrm{E}(|W_k|) &\leqslant \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \frac{(a \log\log n)^2}{\alpha(j,k)(j-i)L} \\ &= \sum_{i=1}^{m-2} \sum_{j=i+1}^{m-1} \frac{(a \log\log n)^2}{(m-j)(j-i)L^2} + \sum_{i=1}^{m-1} \frac{(a \log\log n)^2}{(m-i)L\psi(k)} \\ &\leqslant \left(\frac{a \log\log n}{L}\right)^2 \left(\sum_{t=1}^{m} \frac{1}{t}\right)^2 + \frac{(a \log\log n)^2}{L\psi(k)} \sum_{t=1}^{m-1} \frac{1}{t} \end{aligned}$$

and (11) follows, as does the lemma. □

4. Time Analysis of the Algorithm

The analysis of time complexity of the algorithm PGC is based on

Lemma 3. *The procedure* GREEDY *can be performed in* $O(\log n)$ *time on* PRIORITY *concurrent-read concurrent-write parallel* RAM *with* $O(|V_R|)$ *processors.*

Proof. Lines 2 and 7 of GREEDY can be computed sequentially in constant time. One execution of line 5 needs constant time on PRIORITY concurrent-read concurrent-write parallel RAM if processors $P_1,\ldots,P_r$ proceed as follows

if $\{v_{k_l+k \bmod r}\} \cup X$ is stable **then** $l:=k$;

A possible way how to test whether $\{v\} \cup X$ is stable is to use an array $Stbl(1),\ldots,Stbl(r)$ of boolean variables which are set to **true** when GREEDY is called and to add the next statement to line 6 of GREEDY

for $j:=1$ **to** r **pardo**
if $(j=k_l+l \bmod r)$ **or** (v_j is connected with $v_{k_l+l \bmod r}$)
then $Stbl(j):=$**false**;

Now, $Stbl(j)$ is true iff $v_j \notin X$ and $X \cup \{v_j\}$ is stable. □

An immediate consequence of Lemma 3 is

Theorem 2. *The algorithm* PGC *runs in* $O(\log^2 n \log\log n)$ *worst case parallel time on* PRIORITY *concurrent-read concurrent-write parallel* RAM *with* $O(n^2 \log n^{-2} n)$ *processors.*

5. Conclusions

We have shown that most graphs could be coloured by $O(n \log^{-1} n)$ colours in poly-log worst case parallel time.

The number m of different stable sets which are looked for simultaneously in line 10 of the algorithm is a trade-off between the amount of parallelism (the larger is m the greater is the number of stable sets found in one execution of the loop 7-12) and the amount of overlapping of these sets (the larger is m the greater number of vertices is covered by more than one stable set). The optimal value of m is clearly between $|V_R| \log^{-2} |V_R|$ and $|V_R| \log^{-1} |V_R|$, but it is unknown to the authors.

It would also be interesting to investigate a parallel colouring of sparse random graphs which have large stable subsets, because in such a case our procedure GREEDY does not give maximal stable sets (or, if we remove the test $|X| < 3 \log n$ from line 3 of GREEDY, it would be too slow).

References

[G] Goldschlager, A unified approach to models of synchronous parallel machines, *J. ACM* 29,*A* (1978), 1073–1086.

[GJ] M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, San Francisco, 1979.

[GMcD] G.R. Grimmett, C.J.H. McDiarmid, On colouring random graphs, *Math. Proc. Cambridge Phil. Soc.*, 77 (1975), 313–324.

[Ka] R.M. Karp, Reducibility among combinatorial problems, [in] *Complexity of Computer Computations*, R.E. Miller and J.W. Thatcher, eds., Plenum Press, New York, 1972, 85–104.

[Ku] L. Kučera, Parallel computation and conflicts in memory access, *Information Processing Letters* 14,2 (1982), 93–96.