Parallel Programming Languages & Systems

Exercise Sheet 1
MSc (Level 11) Version

This exercise sheet is assessed and must be your own work. Please be sure that you have read, understood and adhered to the School's guidelines on plagiarism. It accounts for 10% of the course final mark, with a further 10% coming from the second exercise sheet. The best marks will be awarded for simple, correct and clearly argued content. The deadline for the exercise is 4pm, Thursday 16th February, 2012. You should submit your work electronically, as a pdf document using the DICE command

```
submit ppls cw1 pplsEx1.pdf
```

It is important that you use the filename as given, and that your document is in pdf format.

1. Suppose a computer has an atomic decrement (DEC) instruction that also returns a value indicating whether the result is negative, so that it has the effect described by the following pseudocode:

```
bool DEC (int var) =
< var = var-1;
  if (var>=0) return false; else return true; >
```

Using DEC, develop a pseudocode solution to the critical section problem for `n` processes. Don't worry about the "eventual entry" property. Try to make your solution as cache efficient as possible (in the sense of minimising coherence traffic). Describe clearly how your solution works and why it is correct and efficient. Make sure that you have read the course overheads headed "Implementing Locks" (numbers 48-54) before attempting this question.

[You can assume atomic reads and writes and a sequentially consistent model of shared memory. You should also assume that the program runs with an integer type which can store arbitrarily large positive or negative values, so that there is no issue about "wrapping around" when integer arithmetic would otherwise overflow.]

(40 *marks*)

2. Appended to this sheet is an extract from a research paper [1], which describes a parallel algorithm for graph colouring. The algorithm as presented is for a distributed (i.e. message passing) model, with one processor per node of the graph. Your task is to consider and discuss the algorithm in the light of the concepts studied in the course so far, *but for an alternative formulation in which you the algorithm is adapted to run in a shared variable programming context.* Thus, what appear as messages in the given algorithm can be replaced by appropriately chosen and synchronised uses of shared variables in your new version.

   Write a short report (of around a page) on such a shared variable version of the algorithm, discussing, its relationship to any patterns, synchronisation requirements, and the issues which would arise if it were to be amended to allow for more nodes in the graph than processors. *NB You are not being asked to write pseudo-code for the algorithm, and no credit will be given for doing so.*

   The full version of the paper is accessible through the University Library on-line journal collection. However, you do not need to read this to answer the question. It should be enough to read only sections 1 and 2. The remainder of the paper concerns performance analysis (of both run-time and quality of solution).

   (60 *marks*)

References

[1] Hansen J. et al, *Distributed Largest-First Algorithm for Graph Coloring*, Proceedings of EuroPar 2004, Lecture Notes in Computer Science volume 3149, pages 804-811, Springer Verlag, 2004.

Distributed Largest-First Algorithm for Graph Coloring

Jennie Hansen[1], Marek Kubale[2], Łukasz Kuszner[2], and Adam Nadolski[2]

[1] Department of Actuarial Mathematics and Statistics, Heriot-Watt University, Edinburgh EH14 4AS, UK

[2] Department of Algorithms and System Modeling, Gdańsk University of Technology, Narutowicza 11/12, 80-952 Gdańsk, Poland*

Abstract. In the paper we present a distributed probabilistic algorithm for coloring the vertices of a graph. Since this algorithm resembles a largest-first strategy, we call it the distributed LF (DLF) algorithm. The coloring obtained by DLF is optimal or near optimal for numerous classes of graphs e.g. complete k-partite, caterpillars, crowns, bipartite wheels. We also show that DLF runs in $O(\Delta^2 \log n)$ rounds for an arbitrary graph, where n is the number of vertices and Δ denotes the largest vertex degree.

1 Introduction

We discuss the vertex coloring problem in a *distributed network*. Such a network consists of processors and bidirectional communication links between pairs of them. It can be modeled by a graph $G = (V, E)$. The set of vertices V corresponds to processors and the set E of edges models links in the network. To color the vertices of G means to give each vertex a color in such a way that no two adjacent vertices get the same color. If at most k colors are used, the result is called a *k-coloring*.

We assume that there is no shared memory. Each processor knows its own links and its unique identifier. We want these units to compute a coloring of the associated graph without any other information about the structure of G. We assume that the system is synchronized in *rounds*. The number of rounds will be our measure of the time complexity. Such a model of coloring can be used in a distributed multihop wireless network to eliminate packet collisions by assigning orthogonal codes to radio stations [1].

In evaluating the performance of a random distributed coloring algorithm $\mathcal{A}$ on a graph G there are at least two random variables of interest: $C_{\mathcal{A}}(G)$, the number of colors used by the algorithm to color graph G and $T_{\mathcal{A}}(G)$, the number of rounds used to color G.

A good distributed algorithm is one where $C_{\mathcal{A}}(G)$ is close to $\chi(G)$, the chromatic number of G, and where $T_{\mathcal{A}}(G)$ is small relative to the number of vertices

* Supported in part by KBN grants: 4T11C04725 and 4T11C00324.

in G. The difference $C_{\mathcal{A}}(G)-\chi(G)$ can be viewed as a measure of the effectiveness of the algorithm.

We note that it is not always easy to achieve both speed and effectiveness. In [3] a distributed algorithm for $(\Delta+1)$-coloring of graphs was given, where Δ is the largest vertex degree in a graph. Also, analysis of its time complexity was presented. It was proved to run in $O(\log n)$ time. We shall refer to this as the *trivial* algorithm (same as in [2]). The trivial algorithm is extremely simple and fast, however not optimal. In fact, the number of colors used by the algorithm is close to Δ even if the graph is bipartite. This is not surprising as the trivial algorithm has no mechanism of economizing on colors. Further improvements to the trivial algorithm were proposed in [2]. In that paper a new algorithm which is able to compute a coloring using $O(\Delta/\log\Delta)$ colors was given, but it works on triangle-free graphs only and fails on some instances of the problem.

One way to improve the performance of a distributed algorithm is to introduce a strategy into the algorithm which is known to be effective in non-distributive coloring algorithms. For example, it is well known that it is much better to color the vertices in a largest degree first order as in some sequential heuristics like largest-first (LF), smallest-last (SL) and saturation largest-first (SLF) (see e.g. [5]). That is why this strategy is incorporated into the distributed largest-first (DLF) algorithm which was introduced in [6] and which is analyzed thoroughly in this paper.

The paper is organized as follows. In the next section the DLF algorithm is carefully defined. In Section 3 we investigate the time complexity of the DLF algorithm in a special case of regular graphs with fixed degree and in the general case, where no assumptions are made about the structure of G. In the final section we consider the effectiveness of DLF and we show that for some classes of graphs we have $C_{\mathrm{DLF}}(G)-\chi(G)\leq 1$. This holds even when $\chi(G) << \Delta$. In particular, on these classes the DLF algorithm is better than the trivial algorithm in the worst case.

2 DLF Algorithm

In the DLF algorithm each vertex has three parameters:

- its degree in graph: $\deg(v)$,
- random value, which is generated locally and independently: $\mathrm{rndvalue}(v)$,
- palette of forbidden colors, which have already been used by its neighbors: $\mathrm{usedcolor}(v)$ (initially empty).

Within each round the vertices are trying to obtain their colors. If the same color is chosen by neighboring vertices then parameters: $\deg(v)$ and $\mathrm{rndvalue}(v)$ determine the precedence. Specifically, for neighboring vertices $v_1, v_2 \in V$, we say that the priority of v_1 is higher than that of v_2 if:

$$\deg(v_1) > \deg(v_2)$$

or

$$(\deg(v_1) = \deg(v_2)) \text{ and } (\mathrm{rndvalue}(v_1) > \mathrm{rndvalue}(v_2))$$

Within each round every uncolored vertex v executes the following five steps:

1. Choose parameter rndvalue(v) uniformly distributed on [0..1].
2. Send to all neighbors the following parameters: deg(v), rndvalue(v), and the first legal color (not on vertex v's list of forbidden colors).
3. Compare its own parameters with those received from its neighbors and check which vertex has the highest priority.
4. If vertex v's proposed color does not clash with proposals from its neighbors or if v has the highest priority amongst its neighbors, keep the proposed color, send message to neighbors and stop.
5. If not, update list usedcolor(v).

3 Time Complexity

In this section we investigate the complexity of the DLF algorithm. Let $\mathcal{A}$ be a randomized algorithm working on a graph with n vertices. We say that the complexity of $\mathcal{A}$ is $O(F(n))$ if there exist a constant $c > 0$ and $q(n) \in (0..1)$ such that $1/(1-q(n))$ is $O(F(n))$ and the probability that the processing time of $\mathcal{A}$ exceeds $cF(n)+t$ rounds is at most $q(n)^t$, i.e.

$$\Pr[T_{\mathcal{A}}(G) > cF(n)+t] \leq q(n)^t. \tag{1}$$

It is easy to see that if the complexity of $\mathcal{A}$ is $O(F(n))$, then the mean of $T_{\mathcal{A}}(G)$ is $O(F(n))$ as well.

Proposition 1. **[6]** *For any $r-$regular graph G with n vertices the algorithm DLF runs in $O(\log n)$ rounds.*

Proof. Let G be an $r-$regular graph and let v be an uncolored vertex at the beginning of some round k. Let $\Pr(v)$ denote the probability that v gets a color during this round and let $UN(v)$ denote the number of uncolored neighbors of vertex v at the start of round k. Suppose that rndvalue$(v) = x$, then the conditional probability that v gets a color during the k-th round, $\Pr[v \,|\, \text{rndvalue}(v) = x]$, is given by $\Pr[v \,|\, \text{rndvalue}(v) = x] = x^{UN(v)} \geq x^r$. It follows that

$$\Pr(v) \geq \int_0^1 x^r \mathrm{d}x = \frac{1}{r+1}.$$

Using this bound and applying established techniques for analyzing probabilistic recurrence relations (see [4], Theorem 1.1), we obtain

$$Pr[T_{\mathrm{DLF}}(G) \geq \lfloor \log_{(r+1)/r}(n) \rfloor + t + 1] \;\leq\; \left(\frac{r}{r+1}\right)^{t-1}. \;\square$$

3.1 Time Complexity in a General Case

Now we establish a general upper bound on the complexity of the DLF algorithm.