Some Simple Distributed Algorithms for Sparse Networks

Alessandro Panconesi
DSI
Universitá La Sapienza di Roma

Romeo Rizzi
BRICS
University of Århus

October 26, 2000

Abstract

We give simple, deterministic, distributed algorithms for computing maximal matchings, maximal independent sets and colourings. We show that edge colourings with at most $2\Delta - 1$ colours, and maximal matchings can be computed within $\mathcal{O}(\log^* n + \Delta)$ deterministic rounds, where Δ is the maximum degree of the network. We also show how to find maximal independent sets and $(\Delta + 1)$-vertex colourings within $\mathcal{O}(\log^* n + \Delta^2)$ deterministic rounds. All hidden constants are very small and the algorithms are very simple.

Key words: distributed computing, sparse networks, maximal independent set, maximal matching, vertex colouring, edge colouring.

1 Introduction

In this paper, we give fast and simple, deterministic distributed algorithms for computing several graph structures– maximal matchings, maximal independent sets, and vertex– and edge–colourings. The algorithms are very simple and are very fast when the maximum degree of the network is small. In a distributed context, computing these structures quickly can be useful to schedule operations, especially i/o transfers (see [2, 3, 4]). The model we consider is the synchronous, message-passing network. Here, a graph G models a distributed network or architecture, as follows. Every vertex of G corresponds to a processor and every edge to a bidirectional communication link. It is also assumed that every processor has its own unique **ID** and that this is an integer between 1 and n, the number of vertices of G. This is without loss of generality because **ID**s are used only in comparisons with other **ID**s.

Computation proceeds in a sequence of *rounds*, where in each round every processor receives messages from the neighbours, it does some amount of local computation, and sends messages to the neighbours. The complexity of a distributed algorithm, or *protocol*, is, by definition, the number of rounds needed by the algorithm to compute. Since typically sending messages is orders of magnitude more costly than performing a "reasonable" amount of local computation, this model gives a rough, but reasonably good, approximation of the cost incurred by distributed protocols. Notice that in particular the cost of sending a message between two nodes must be at least proportional to their distance in the network. This makes the model orthogonal to the PRAM where communication is completely free and only computation is charged for. As stated, computation is free in our model but if needed it can be easily taken into account– just charge for it! We remark that the algorithms described in this paper perform very simple local steps and therefore their cost, including computation, is the same order of magnitude as the stated communication cost. Our results are as follows. The input to the algorithms is a (distributed) network of maximum degree Δ and n vertices.

- We give an $\mathcal{O}(\Delta + \log^* n)$ algorithm for computing maximal matchings;
- We give an $\mathcal{O}(\Delta + \log^* n)$ algorithm for computing $(2\Delta - 1)$-edge colourings;
- We give an $\mathcal{O}(\Delta^2 + \log^* n)$ algorithm for computing $(\Delta + 1)$-vertex colourings;
- We give an $\mathcal{O}(\Delta^2 + \log^* n)$ algorithm for computing maximal independent sets.

We remark that the hidden constants here are really small. This makes our algorithms "local" in the following sense: if we keep Δ fixed and let n, the number of vertices, grow, the complexity remains essentially constant.

Comparison with previous work. While maximal matchings can be computed in polylogarithmic, in n, time in the distributed model [8], it is a decade old open problem whether the same running time is achievable for the remaining 3 structures [1, 8, 9, 10]. The maximal matching algorithm in [8] takes $\mathcal{O}(\log^4 n)$ rounds and therefore this result appears to be at the moment only of theoretical interest. For bounded degree graphs the situation looks somewhat better. In particular, Goldberg and Plotkin [5] give algorithms for the problems we consider whose complexity is $\mathcal{O}(\log^* n)$. At first glance this looks better than the complexity of our algorithms, but there is a catch. There is a hidden additive constant which is at least Δ^Δ where Δ is the maximum degree of the network. Therefore our algorithms compare favourably with those in [5].

In a short but interesting paper, Linial [9] showed that $\Omega(\log^* n)$ many communication rounds is a lower bound for computing the graph structures considered in this paper on a ring topology. This result *does not* imply the same bound for all constant degree graphs. Intuitively, low degrees "decrease" the capability of the network to disperse information quickly. The result however does easily generalize to constant degree graphs (say, Δ-regular graphs, for Δ constant) for maximal independent sets and $(\Delta + 1)$-vertex colouring (just replace every edge of the ring with a Δ-clique). It does not seem to generalize so easily to maximal matchings and $(2\Delta - 1)$-edge colourings and we leave this as an open problem. In the same paper, Linial also showed that within $\mathcal{O}(\log^* n)$ many communication rounds it is possible to compute vertex colourings using $\mathcal{O}(\Delta^2)$ many colours.

In [1], a very simple, deterministic vertex colouring algorithm is given whose complexity is $\mathcal{O}(\Delta \log n)$ many rounds. The algorithm uses $\Delta + 1$ many colours (and can be used to edge colour the network with $2\Delta - 1$ colours). With a straightforward modification the algorithm also computes maximal independent sets (and maximal matchings). Therefore, as far as maximal independent sets and vertex colourings are concerned, this algorithm is still asymptotically better than our algorithms for values of Δ larger than $\log n$. The simplicity of the algorithm in [1] is comparable to that of our algorithms.

2 Generating a Forest Decomposition in Constant Time

Our algorithms are based on a simple procedure which partitions the edge set of the input network in constant time. The decomposition is generated as follows:

- Let d_u be the degree of vertex u. Each vertex u, in parallel, ranks the edges incident upon itself arbitrarily. By ranking we mean that each edge incident upon u receives a distinct number between 1 and d_u. We call this number u's *proposal* for that edge. Therefore every edge gets two proposals, one for each endpoint.

- The *colour* of edge uv is defined to be the proposal of the endpoint with highest **ID**.

This procedure partitions the edge set into at most Δ classes. We now show that each class is a forest of rooted arborescences. Recall that a *rooted arborescence* is a rooted, directed tree.

Claim 2.1 *For $i = 1, \ldots, \Delta$ let E_i be the set of edges with colour i. Then $F_i := (V, E_i)$ is a forest of G, for $i = 1, \ldots, \Delta$. Furthermore, if every edge is oriented from the endnode of smaller* **ID** *to the one of higher* **ID**, *then all such forests consist of outward rooted arborescences.*

Proof: First we observe that no two edges of the same colour can be oriented towards a same node, say v. Assume to the contrary that u_1v and u_2v are two such edges. But v assigns different proposals to the edges incident to itself, a contradiction.

To conclude the proof we must exclude the existence of directed cycles inside any F_i. Now, since every edge is oriented from the endnode of smaller **ID** to the one of higher **ID**, we can exclude the existence of directed cycles altogether, since the orientation is induced by a total order of the nodes. □

Definition 2.2 *We shall refer to the forest so computed as a* forest decomposition of *G*.

The algorithm works with an arbitrary proposal scheme that orders the edges incident to a vertex. In particular, the following Algorithm 1 can be adopted:

Algorithm 1 ROOTED_TREES

1. Each vertex sends its **ID** number to all of its neighbors;
2. The edges incident upon each vertex are ordered by decreasing **ID** value of the other endnode; the rank so obtained is the proposal made by the vertex;
3. Each edge selects the proposal coming from the endnode of higher **ID**.

3 Matchings and Edge Colourings

In [5], Goldberg et al. give a distributed algorithm to colour the nodes of a rooted arborescence T with three colours C_1, C_2, C_3 within $\mathcal{O}(\log^* n)$ rounds, provided that each vertex has its own **ID**, knows its father, and the root knows to be the root. The hidden constant here is very small. The algorithm was devised for the PRAM model but it is easily verified to be a bonafide distributed algorithm in our sense. The arborescences of Claim 2.1 satisfy the above proviso and therefore the procedure of Goldberg et al. can be applied to them.

We start by showing how to compute a maximal matching in a directed tree T within $\mathcal{O}(\log^* n)$ rounds. Let T be a rooted arborescence whose nodes are partitioned into three disjoint independent sets C_1, C_2, C_3. If one just chooses a single outgoing edge for every node in C_1, then the edges so selected are independent. Let M_1 be this matching and repeat the operation, now with C_2, in the "left over" subtree obtained by removing all nodes in M_1. This will return a matching M_2 which can be added to M_1, and, repeating with C_3, a last matching M_3 is obtained. Let us now show that the resulting edge set $M := M_1 \cup M_2 \cup M_3$ is a maximal matching. First notice that when M_i is computed, the edges selected by the vertices of colour i do not share endpoints currently in the tree since only outgoing edges are selected. Second, these edges cannot share endpoints with edges included in a previous M_j, $j < i$, because all vertices matched by M_j were removed from the tree. Therefore, $M := M_1 \cup M_2 \cup M_3$ is a matching. To see that M is maximal, suppose by contradiction that an edge uv could be added to M. Suppose moreover without loss of generality that u is the father of v and that $C(u) = i \neq j = C(v)$. Let us consider the time when colour i is processed, i.e., when M_i is computed. There are two cases. The first is that the edge uv is not present in the tree at that moment. This implies that either u or v has been previously matched, a contradiction. The other case is that u is present. But then the set of edges outgoing from u is nonempty, which implies that u will be matched. Again, a contradiction. Therefore M is maximal.

Since the trees comprising each forest are vertex disjoint, a maximal matching in a forest can be computed by simultaneously computing maximal matchings in each tree of the forest, and by adding these matchings together. We have established the following.

Fact 3.1 *Let F be a forest of the forest decomposition of G. If F is 3-vertex coloured then, a maximal matching of F can be computed within 3 communication rounds.*

Therefore, denoting by $F_1, \ldots, F_\Delta$ a forest decomposition of G, and assuming that each forest is already 3-vertex coloured, a maximal matching $M_1 \cup M_2 \cup \ldots \cup M_\Delta$ is obtained as follows. First, compute a maximal matching M_1 in F_1, delete the vertices of M_1 from G, thereby obtaining a "left-over" graph G' together with a "left-over" forest decomposition $F'_2, \ldots, F'_\Delta$. Then, compute M_2 in F'_2, remove all nodes in M_2, and so on, for a total of Δ such phases. The algorithm is spelled out as Algorithm 2.

Clearly, this procedure computes a maximal matching. As noted, computing a 3-vertex colouring of the forests takes $\mathcal{O}(\log^* n)$ many rounds, since this can be done simultaneously for all F_i. Building the matching incrementally takes Δ phases, each of which necessitates $\mathcal{O}(1)$ many rounds.

Theorem 3.1 *A maximal matching of G can be computed within $\mathcal{O}(\log^* n + \Delta)$ rounds.*

Let us now switch to the problem of computing an edge colouring of G with $2\Delta - 1$ colours within $\mathcal{O}(\log^* n + \Delta)$ rounds. One possibility would be to compute in sequence $2\Delta - 1$ maximal

Algorithm 2 MATCH

1. Compute a forest decomposition $F_1, \ldots, F_\Delta$ of G.
   Direct all edges from lower **ID** node to higher **ID** node;
2. Compute a 3-vertex colouring of each F_i, in parallel.
   Let c_i be the colouring of F_i;
3. $\mathcal{M} := \emptyset$;
4. for $i := 1$ to Δ do
5.   for $c := 1$ to 3 do
6.   Every u such that $c_i(u) = c$ selects arbitrarily one of its outgoing edges;
         let M_c be the set of edges so selected;
7. $\mathcal{M} := \mathcal{M} \cup M_c$;
8.   Remove all vertices of M_c from the graph.

matchings, but this would take $\Omega(\Delta^2 + \log^* n)$ many rounds. We shall then proceed as follows. Let F_i be a forest of a forest decomposition of G and assume that it is 3-coloured already. Let $c : V(F_i) \to \{1, 2, 3\}$ be the 3-colouring. Consider the following partition of $E(F_i)$, the edge set of F_i, into three sets E_i^c, $c = 1, 2, 3$, where

$$E_i^c := \{uv : u \text{ is the tail and } c(u) = c\}.$$

Observation 3.2 *Each set E_i^c is made of node-disjoint stars whose centers are the nodes of F_i of colour c.*

Note that the sets E_i^c partition E. Therefore, Algorithm 3, whose correctness will be argued shortly, computes a $(2\Delta - 1)$-edge colouring.

Algorithm 3 EDGE_COLOR

1. Compute a forest decomposition $F_1, \ldots, F_\Delta$ of the input graph G and a 3-vertex colouring c_i for each F_i;
2. for $i = 1, \ldots, \Delta$ do:
3.   for $c_i = 1, 2, 3$ do:
4.   The centers of the stars of E_i^c assign different colours from the interval $[1, \ldots, 2\Delta - 1]$ to the edges in their respective stars, paying attention not to create conflicts with previously coloured edges.

To convince ourselves that Step 4 of the above algorithm can always be carried out, recall that each edge is adjacent to at most $2\Delta - 2$ other edges, and that we are using $2\Delta - 1$ colours. Observation 3.2 implies, for given i and c, that the colouring operations of the stars are always mutually compatible. Therefore the algorithm computes a $(2\Delta - 1)$-edge colouring. The complexity is clearly as stated.

Theorem 3.2 *Algorithm 3 computes a $(2\Delta - 1)$-edge colouring of the input graph G in $\mathcal{O}(\Delta + \log^* n)$ many communication rounds.*

4 Vertex Colourings and Maximal Independent Sets

In this section we show how to find a proper $(\Delta + 1)$-colouring of the nodes of G within $\mathcal{O}(\log^* n + \Delta^2)$ deterministic rounds. As a consequence, we will obtain a deterministic protocol to find a maximal independent set of G within the same bound. Note that the complement of a maximal independent set is a minimal node cover and also a minimal dominating set. Therefore, our algorithm applies to these problems as well.

The vertex colouring algorithm is based on the following two ideas. The first idea is that if G is k-vertex coloured, where perhaps k is much larger than Δ, the maximum degree of G, the k-colouring can be shrunk to a $(\Delta + 1)$-vertex colouring simply as follows: For each $i = \Delta + 2, \Delta + 3, \ldots, k$, all vertices with colour i, in parallel, recoulor by picking any available colour in the set $\{1, \ldots, \Delta + 1\}$. This is correct since each colour class of the original k-colouring is an independent set and therefore all recolouring choices are mutually compatible. The number of rounds needed is $k - \Delta - 1$. We shall refer to this as the *shrinking procedure*.

The second idea is that the problem of $(\Delta + 1)$-vertex colouring G can be reduced to that of 3-vertex colouring each forest of a forest decomposition of G, as follows. Let $F_1, \ldots, F_\Delta$ be the forest decomposition, and let

$$A := \bigcup_{i=1}^{\ell} E(F_i)$$

be the edge set of the first ℓ forests. Suppose, by induction, that $G[A]$, the subgraph induced by the edge set A, is $(\Delta + 1)$-vertex coloured already, and that $F_{\ell+1}$ is 3-vertex coloured. Let χ and c, respectively, be the two colourings of $G[A]$ and $F_{\ell+1}$. Then, (χ, c) is a $(3\Delta + 3)$-colouring of $G[A \cup F_{\ell+1}]$ which can be shrunk to a $(\Delta + 1)$-colouring in $2\Delta + 2$ rounds by the shrinking procedure. Here is the resulting Algorithm 4.

Algorithm 4 COLOR

---

1. Compute a forest decomposition $F_1, \ldots, F_\Delta$ of the input graph G and a 3-vertex colouring c_i for each F_i;
2. for $i = 1, \ldots, \Delta$ do $\qquad$ $\{$include F_i one at a time$\}$
3. $\quad$ for $k := 1$ to $\Delta + 1$ do $\qquad$ $\{$shrink (χ, c) to a $(\Delta + 1)$ colouring $\}$
4. $\quad\quad$ for $c := 2$ to 3 do
5. $\quad\quad\quad$ all vertices u such that $\chi(u) = k$ and $c_i(u) = c$ set $\chi(u)$ to an arbitrary colour in the set $\{1, \ldots, \Delta + 1\}$ and not assigned to any neighbor v of u with $c_i(v) < c$.

To obtain a maximal independent set we only need to consider a minor modification of Algorithm 4. In step 5, instead of assigning u to any available colour, assign u to the smallest possible colour. Then, when Algorithm 4 terminates colour class 1 will be a maximal independent set of G. Clearly, each colour class computed by Algorithm 4 is an independent set. Maximality follows since if $\chi(u) \neq 1$ then u has at least one neighbour with colour 1.

Remark: These algorithms will work well even if a small portion of the vertices has degree higher than a parameter Δ because they can colour themselves after the bulk of the graph has coloured itself. This might lead to very efficient algorithms when the network has only few high degree nodes.

Acknowledgement

All authors gratefully acknowledge the hospitality of BRICS, Århus, Denmark, where this research was performed. We also thank Riccardo Silvestri and Aravind Srinivasan for useful conversations.

References

[1] B. Awerbuch, A.V. Goldberg, M. Luby, and S.A. Plotkin, Network decomposition and locality in distributed computation. In 30th Annual Symposium on Foundations of Computer Science, pages 364-369, November 1989. IEEE

[2] R Jain, J Werth, J.C. Browne, and G Sasaki, *A graph-theoretic model for the scheduling problem and its application to simultaneous resource scheduling*, in **Computer Science and Operations Research: New Developments in Their Interfaces**, Ed. by O. Balci, R. Shander, and S. Zerrick, Penguin Press, 1992.

[3] R Jain, K. Somalwar, J Werth and J.C. Browne, *Scheduling Parallel I/O Operations in Multiple Bus Systems*, Journal of Parallel and Distributed Computing, 16(4), pp. 352-362, 1992.

[4] R Jain and J Werth *Analysis of Approximate Algorithms for Constrained and Unconstrained Edge Coloring of Bipartite Graphs,* DIMACS Technical Report, 95-01 January, 1995, Appeared in Information Processing Letters,

[5] A. Goldberg and S.A. Plotkin, Parallel $(\Delta+1)$-coloring of constant-degree graphs. Inform. Process. Lett., 25 (1987), no. 4, 241–245.

[6] A. Goldberg, S. Plotkin and G.E. Shannon, Parallel symmetry breaking in sparse graphs SIAM J. Disc. Math. Vol. 1, No. 4, pp. 434-446, November 1988

[7] A. Goldberg, M. Luby, S. Plotkin and G.E. Shannon, Parallel symmetry breaking in sparse graphs SIAM J. Disc. Math. Vol. 1, No. 4, pp. 434-446, November 1988

[8] M. Hanckowiak, M. Karonski and A. Panconesi, A faster distributed algorithm for computing maximal matchings deterministically, in Proceedings of PODC 99, the Eighteenth Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing.

[9] N. Linial, Locality in distributed graph algorithms, SIAM J. Comput., Vol. 21, No. 1, pp. 193-201, February 1992

[10] N. Linial and M. Saks, Low diameter graph decompositions, Combinatorica (1993), Vol. 13 (4)