CS 590R–Spring 2003
Algorithms for Communications Networks
Professor : Gopal Pandurangan
Notes by : Mohamed R. Fouad
Lecture : #5

Distributed Algorithms

1 Definition

Distributed algorithms, DAs, are algorithms that were designed to run on many processors "distributed" over a large geographical area. DAs are extremely important in network applications in which there is no central processing unit but rather each node executes the algorithm independently. In addition, DAs can run over LANs, ad–hoc networks, and peer–to–peer networks where there is a very little infrastructure[1].

There are many different kinds of DAs:

- Interprocess communication methods: this includes accessing shared memory, point-to-point or broadcast messages, and remote procedure calls.
- Timing models: this includes synchronous/asynchronous models. In asynchronous models, there are no assumptions about the timings of the operations and they can be interleaved in an arbitrary manner. On the contrary, synchronous models are much easier to manage but at the expense of setting up a global clock to coordinate between different operations. It is almost similar to a parallel computation in the sense that it operates under a "lock-step" fashion.
- Failure models: reliable or faulty behavior; Byzantine failures (failed processor can behave arbitrarily).
- Various problems: such as resource allocation, communication, consensus, concurrency control, deadlock detection, and many traditional algorithms.

[1]Unlike the networks of ISPs, where the infrastructure is well established beforehand.

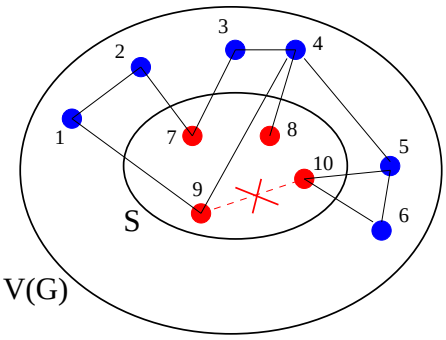

Figure 1: An example showing a maximal independent set S of a graph G

2 Local Algorithms

A distributed algorithm is a *localized algorithm* if it uses only the information of all the local nodes (for ex., the IP-address of its neighbors) plus a small amount (preferably a constant) of global information about other nodes. It is very crucial that we do not only have a distributed algorithm but also a local algorithm.

3 An Example

In this section we present and analyze one of the distributed local algorithms, the *maximal independent set, MIS*.

3.1 The Maximal Independent Set Problem

A set of nodes is called an *independent set* if it contains no pair of adjacent nodes, and it is *maximal* if it cannot be increased to form a larger independent set by the addition of other nodes.

In Figure 1, $S = \{7, 8, 9, 10\}$ is one of the maximal independent sets of the graph G since we don't have an edge connecting any pair of these vertices. Moreover, you may notice that each vertex outside S is connected to at least one of the vertices inside S or, otherwise, that vertex should be included in S. Therefore, it is not possible to add more vertices to S without violating the MIS property. You may also notice that S is not the *maximum* independent set[2], S^*. Replacing vertex 7 with vertices 2 and 3 would lead to another maximal independent set $S' = \{2, 3, 8, 9, 10\}$, where $|S'| > |S|$.

The MIS problem has various important applications:

[2] $|S^*| = \max_i |S_i|$. Identifying S^* is an NP-complete problem.

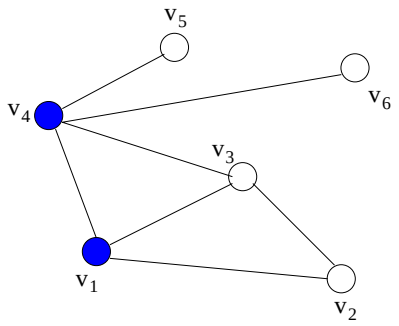

Figure 2: The dominating set is not necessarily an independent set

- Resource allocation: The system resources are allocated in such a way that no two neighboring processes should work simultaneously.

- Dominating sets: The set $V' \subseteq V$ is called a *dominating set* if for all vertices $v \notin V'$, there exists at least one vertex $u \in V'$ where u and v are adjacent (neighbors). In other words, a dominating set for a graph is a set of vertices whose neighbors, along with themselves, constitute all the vertices in the graph. From this definition, it follows easily that any maximal independent set is also a dominating set but not vice versa. In Figure 2, the set $S = \{v_1, v_4\}$ is a dominating set since the other vertices are connected to at least one of the vertices of S. However, S is not an independent set since v_1 and v_4 are adjacent. On the contrary, the set S in Figure 1 is both a maximal independent set and a dominating set.

- Clusterhead selection: A set of *clusterheads* is a subset of nodes which are basically the major processing centers and should be able to communicate with some other nodes in the neighborhood to collect information. Too many clusterheads would lead to a much more complicated network while too few clusterheads might lead to disconnectivity problems.

The MIS problem is described as follows: given an arbitrary network $G = (V, E)$, we want to design a distributed (and local) algorithm which will output an MIS set in the sense that every node will know whether it is in the MIS or not.

A straight forward algorithm is to pick up a vertex at random and put it in a set S, then we delete all its neighbors. The process continues until there are no more vertices remaining. The complexity of this algorithm would be of $O(m)$ where m is the number of edges in the graph. In addition, because

of the sequential nature of this algorithm in which each iteration depends on its predecessors, it is very hard to apply this algorithm in a distributed fashion. In the subsequent sections, we will show that the algorithm can run in $O(\log n)$ rounds when implemented in a distributed fashion, where n is the number of vertices in the graph.

3.2 Algorithm

Let $\Gamma(v)$ be the set of vertices in V that are adjacent to v. The basic idea is that the algorithm proceeds in rounds; in every round it finds an independent set S, adds S to I (initially I is empty) and deletes $S \cup \Gamma(S)$ from the graph. Where $\Gamma(S) = \bigcup_{v \in S} \Gamma(v)$. Moreover, the degree of a vertex v is $d(v) = |\Gamma(v)|$.

1. $I = \phi$
2. **repeat**
   - 2.1 **for** all $v \in V$ **do**
     - **if** $d(v) = 0$
       - **then** add v to I and delete v from V
       - **else** mark v with probability $1/(2d(v))$
   - 2.2 **for** all $(u, v) \in E$ **do**
     - **if** both u and v are marked
       - **then** unmark the lower degree vertex
   - 2.3 **for** all $v \in V$ **do**
     - **if** v is marked **then** add v to S.
   - 2.4 $I = I \cup S$
   - 2.5 delete $S \cup \Gamma(S)$ from V, and all incident edges from E
3. **until** $V = \phi$

3.3 Remarks

1. The algorithm assumes nothing about the graph (i.e. it runs over arbitrary graphs).

2. Each node executes each phase independently in a "lock-step" fashion. It is assumed that there is a global synchronization clock to regulate

transitions to different phases.

3. Each node knows n and the degree of its neighbors [3].

4. Step 2.1 shows a biasing towards low-degree vertices. The intuition behind this is that they are less likely to be adjacent to each others in the beginning of each round.

5. Step 2.2 shows a biasing towards high-degree vertices. The reasoning behind this is to delete as many edges as possible.

3.4 Analysis

We will show that the expected fraction of edges removed from E during each iteration is bounded from below by a constant.

A vertex $v \in V$ is "good" if it has at least $d(v)/3$ neighbors of degree no more than $d(v)$; otherwise the vertex is "bad". An edge is "good" if at least one of its endpoints is a good vertex, and it is "bad" if both endpoints are bad vertices.

We will show that a good vertex is likely to have one of its lower degree neighbors in S and thereby deleted from V.

Lemma 3.1 *Let $v \in V$ be a good vertex with degree $d(v) > 0$. Then, the probability that some vertex $w \in \Gamma(v)$ gets marked is at least $1 - e^{-1/6}$.*

Proof: Each vertex $w \in \Gamma(v)$ is marked independently with probability $1/(2d(w))$. Let w be selected such that $d(w) \leq d(v)$.

The probability that w is unmarked is

$$\left(1 - \frac{1}{2d(w)}\right) \leq \left(1 - \frac{1}{2d(v)}\right)$$

The probability that none of the neighbors of v gets marked is at most

$$\left(1 - \frac{1}{2d(v)}\right)^{d(v)/3} \leq e^{-\frac{1}{2d(v)}\frac{d(v)}{3}} = e^{-1/6}$$

□

The above lemma shows that if v is a good vertex, there is a good chance[4] that one of its neighbors gets marked in phase 2.1 and causes v to be deleted.

[3] This can be done by sending local messages.

[4] With probability more than $(1 - e^{-1/6})$.

Lemma 3.2 *During any round, if a vertex w is marked, then it is selected to be in S with probability at least $1/2$.*

Proof: The only reason a marked vertex w becomes unmarked is that one of its neighbors $u_i \in \Gamma(w)$ of degree $d(u_i) \geq d(w)$ is also marked. The probability of this happening is $\frac{1}{2d(u_i)} \leq \frac{1}{2d(w)}$. The probability that at least one of the vertices v_i gets marked is

$$Pr\{\bigcup_{u_i \in \Gamma(w)} u_i \text{ is marked}\} \leq \sum_{u_i \in \Gamma(w)} \frac{1}{2d(u_i)} \leq \sum_{u_i \in \Gamma(w)} \frac{1}{2d(w)} \leq \frac{d(w)}{2d(w)} = \frac{1}{2}$$

□

The above lemma shows that there is a good chance that the neighbors of v stay marked in phase 2.2.

Lemma 3.3 *The probability that a good vertex belongs to $S \cup \Gamma(S)$ is at least $(1 - e^{-1/6})/2$.*

Proof: By combining the previous lemmas, the good vertex belongs to $S \cup \Gamma(S)$ if at least one of its neighbors is marked and will stay marked. Thus, the result of this lemma is the product of the results of the previous lemmas.

□

Lemma 3.4 *In a graph $G = (V, E)$ the number of good edges is at least $|E|/2$.*

Proof:

Direct the edges in E from the lower degree end-point to the higher degree end-point breaking ties arbitrarily. Let $d_i(v)$ and $d_o(v)$ be the in-degree and out-degree of v, respectively.

For each bad vertex v,

$$d_o(v) - d_i(v) \geq d(v)/3 = \frac{d_o(v) + d_i(v)}{3}$$

Let $E(S, T)$ be the set of edges directed from vertices in S to vertices in T; and $e(S, T) = |E(S, T)|$. The total degree of the bad vertices is given by

$$
\begin{aligned}
2e(V_B,V_B)+e(V_B,V_G)+e(V_G,V_B) &= \sum_{v\in V_B}(d_o(v)+d_i(v)) \\
&\leq 3\sum_{v\in V_B}(d_o(v)-d_i(v)) \\
&= 3\sum_{v\in V_G}(d_i(v)-d_o(v)) \\
&= 3[(e(V_B,V_G)+e(V_G,V_G)) \\
&\quad -(e(V_G,V_B)+e(V_G,V_G))] \\
&= 3[e(V_B,V_G)-e(V_G,V_B)] \\
&\leq 3[e(V_B,V_G)+e(V_G,V_B)]
\end{aligned}
$$

Thus,

$$
\begin{aligned}
2e(V_B,V_B)+e(V_B,V_G)+e(V_G,V_B) &\leq 3[e(V_B,V_G)+e(V_G,V_B)] \\
2e(V_B,V_B) &\leq 2[e(V_B,V_G)+e(V_G,V_B)] \\
e(V_B,V_B) &\leq e(V_B,V_G)+e(V_G,V_B)
\end{aligned}
$$

Thus, the number of good edges $\geq$ the number of bad edges and, therefore at least 1/2 of the total number of edges are good edges.

□

Now, let

$$
X_i = \begin{cases} 1 & \text{if edge } E_i \text{ is good} \\ 0 & \text{if edge } E_i \text{ is bad} \end{cases}
$$

$E(X_i) = Pr\{X_i = 1\} = Pr\{E_i \; is \; good\} \geq \frac{1-e^{-1/6}}{2}$ from lemma 3.3. By linearity of expectation it follows that the constant factor which gets pruned each round is :

$$
\sum_{i=1}^{|E|/2} E(X_i) \geq \frac{|E|}{2}\frac{1-e^{-1/6}}{2}
$$

4 Next class

The analysis we presented earlier needs to emphasize on the case of probabilistic pruning of edges. In this case, we cannot say that each round we throw away a constant fraction of edges. Rather, this fraction should be treated as a random variable with some distribution. In next class, the random pruning will be modeled and analyzed as a particle moving at random from position n until reaching position 1 according to a certain distribution.