CSE 638: Advanced Algorithms

Lectures 14 & 15
(Parallel Maximal Independent Set)

Rezaul A. Chowdhury
Department of Computer Science
SUNY Stony Brook
Spring 2013

Independent Sets

Let $G = (V, E)$ be an undirected graph.

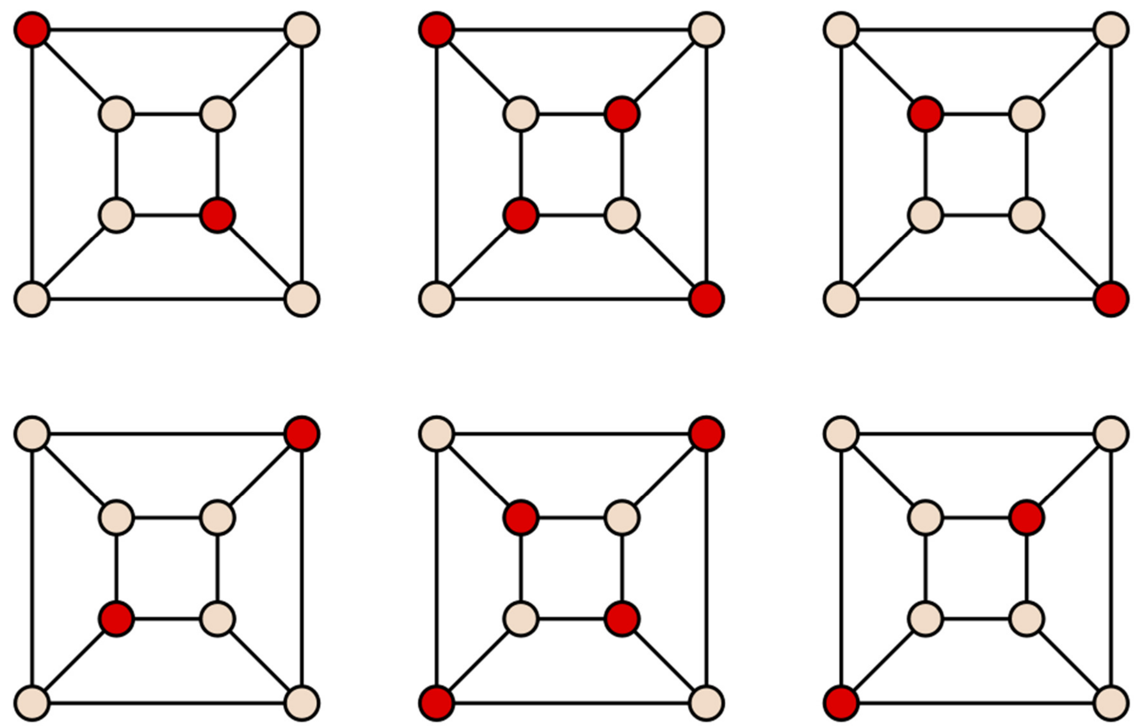
Independent Set: A subset $I \subseteq V$ is said to be *independent* provided for each $v \in I$ none of its neighbors in G belongs to I.

Maximal Independent Set: An independent set of G is *maximal* if it is not properly contained in any other independent set in G.

Maximum Independent Set: A maximal independent set of the largest size.

Finding a maximum independent set is NP-hard.

But finding a maximal independent set is trivial in the sequential setting.



Maximal Independent Sets (red vertices) of the Cube Graph
Source: Wikipedia

Finding a Maximal Independent Set Sequentially

Input: *V* is the set of vertices, and *E* is the set of edges. For each $v \in V$, we denote by $\Gamma(v)$ the set of neighboring vertices of *v*.

Output: A maximal independent set *MIS* of the input graph.

```
Serial-Greedy-MIS ( V, E )
1. MIS ← φ
2. for v ← 1 to |V| do
3.     if MIS ∩ Γ( v ) = φ then MIS ← MIS ∪ { v }
4. return MIS
```

This algorithm can be easily implemented to run in $\Theta(n + m)$ time, where n is the number of vertices and m is the number of edges in the input graph.

The output of this algorithm is called the *Lexicographically First MIS* (LFMIS).

Finding a Maximal Independent Set Sequentially

Input: V is the set of vertices, and E is the set of edges. For each $v \in V$, we denote by $\Gamma(v)$ the set of neighboring vertices of v.

Output: A maximal independent set *MIS* of the input graph.

```
Serial-Greedy-MIS-2 ( V, E )
MIS ← φ
while |V| > 0 do
    pick an arbitrary vertex v ∈ V
    MIS ← MIS ∪ { v }
    R ← { v } ∪ Γ( v )
    V ← V \ R
    E ← E \ { ( v1, v2 ) | v1 ∈ R or v2 ∈ R }
return MIS
```

Always choosing the vertex with the smallest id in the current graph will produce exactly the same MIS as in *Serial-Greedy-MIS*.

Finding a Maximal Independent Set Sequentially

Input: V is the set of vertices, and E is the set of edges. For each $S \subseteq V$, we denote by $\Gamma(S)$ the set of neighboring vertices of S.

Output: A maximal independent set *MIS* of the input graph.

```
Serial-Greedy-MIS-3 ( V, E )
MIS ← φ
while |V| > 0 do
    find an independent set S ⊆ V
    MIS ← MIS ∪ S
    R ← S ∪ Γ( S )
    V ← V \ R
    E ← E \ { ( v1, v2 ) | v1 ∈ R or v2 ∈ R }
return MIS
```

Parallelizing Serial-Greedy-MIS-3

— Number of iterations can be kept small by finding in each iteration an S with large $S \cup \Gamma(S)$. But this is difficult to do.

— Instead in each iteration we choose an S such that a large fraction of current edges are incident on $S \cup \Gamma(S)$.

```
Serial-Greedy-MIS-3 ( V, E )
1. MIS ← φ
2. while |V| > 0 do
3.     find an independent set S ⊆ V
4.     MIS ← MIS ∪ S
5.     R ← S ∪ Γ( S )
6.     V ← V \ R
7.     E ← E \ { ( v1, v2 ) | v1 ∈ R or v2 ∈ R }
8. return MIS
```

— To select S we start with a random $S' \subseteq V$.

- By choosing lower degree vertices with higher probability we are likely to have very few edges with both end-points in S'.
- We check each edge with both end-points in S', and drop the end-point with lower degree from S'. Our intention is to keep $\Gamma(S')$ as large as we can.
- After removing all edges as above we are left with an independent set. This is our S.
- We will prove that if we remove $S \cup \Gamma(S)$ from the current graph a large fraction of current edges will also get removed.

Randomized Maximal Independent Set (MIS)

Input: n is the number of vertices, and for each vertex $u \in [\, 1, n \,]$, $V[\, u \,]$ is set to u. E is the set of edges sorted in non-decreasing order of the first vertex. For every edge $(\, u, v \,)$ both $(\, u, v \,)$ and $(\, v, u \,)$ are included in E.

Output: For all $u \in [\, 1, n \,]$, $MIS[\, u \,]$ is set to 1 if vertex u is in the MIS.

```
Par-Randomized-MIS ( n, V, E, MIS )
while |V| > 0 do
    array d[ 1 : |V| ], c[ 1 : |V| ] = { 0 }, M[ 1 : |V| ] = { 0 }
    parallel for i ← 1 to |E| do
        if i = |E| or E[ i ].u ≠ E[ i + 1 ].u then c[ E[ i ].u ] ← i
    parallel for u ← 1 to |V| do
        if u = 1 then d[ u ] ← c[ u ] else d[ u ] ← c[ u ] - c[ u - 1 ]
        if d[ u ] = 0 then M[ u ] ← 1
        else M[ u ] ← 1 ( with probability 1 / ( 2d[ u ] ) )
    parallel for each ( u, v ) ∈ E do
        if M[ u ] = 1 and M[ v ] = 1 then
            if d[ u ] ≤ d[ v ] then M[ u ] ← 0 else M[ v ] ← 0
    parallel for u ← 1 to |V| do
        if M[ u ] = 1 then MIS[ V[ u ] ] ← 1
    ( V, E ) ← Par-Compress ( V, E, M )
```

$d[\, u \,]$ (i.e., degree of vertex u) can now be computed easily by subtracting $c[\, u - 1 \,]$ from $c[\, u \,]$

if both end-points of an edge is marked, unmark the one with the lower degree

remove marked vertices along with their neighbors as well as the corresponding edges

for each u find the edge with the largest index i such that $E[\, i \,].u = u$, and store that i in $c[\, u \,]$

mark lower-degree vertices with higher probability

add all marked vertices to MIS

Removing Marked Vertices and Their Neighbors

Input: Arrays V and E, and bit array $M[\,1{:}\ |V|\,]$. Each entry of E is of the form $(\,u, v\,)$, where $1 \le u, v \le |V|$. If for some u, $M[\,u\,] = 1$, then u and all v such that $(\,u, v\,) \in E$ must be removed from V along with all edges $(\,u, v\,)$ from E.

Output: Updated V and E.

```
Par-Compress ( V, E, M )
array S_V[ 1 : |V| ] = { 1 }, S'_V[ 1 : |V| ], S_E[ 1 : |E| ] = { 1 }, S'_E[ 1 : |E| ]
parallel for u ← 1 to |V| do
    if M[ u ] = 1 then S_V[ u ] ← 0
parallel for i ← 1 to |E| do
    u ← E[ i ].u, v ← E[ i ].v
    if M[ u ] = 1 or M[ v ] = 1 then S_V[ u ] ← 0, S_V[ v ] ← 0, S_E[ i ] ← 0
S'_V ← Par-Prefix-Sum ( S_V, + ), S'_E ← Par-Prefix-Sum ( S_E, + )
array U[ 1 : S'_V[ |V| ] ], F[ 1 : S'_E[ |E| ] ]
parallel for u ← 1 to |V| do
    if S_V[ u ] = 1 then U[ S'_V[ u ] ] ← V[ u ]
parallel for i ← 1 to |E| do
    if S_E[ i ] = 1 then F[ S'_E[ i ] ] ← E[ i ]
parallel for i ← 1 to |F| do
    u ← F[ i ].u, v ← F[ i ].v
    F[ i ].u ← S'_V[ u ], F[ i ].v ← S'_V[ v ]
return ( U, F )
```

initialize

marked vertices will be removed

neighbors of marked vertices & corresponding edges must go

find new indices for surviving vertices & edges

move surviving vertices to the smaller array U

move surviving edges to the smaller array F

update the end-points of the surviving edges to new vertex indices

Removing Marked Vertices and Their Neighbors

```
Par-Compress ( V, E, M )
1.  array S_V[ 1 : |V| ] = { 1 }, S′_V[ 1 : |V| ],
          S_E[ 1 : |E| ] = { 1 }, S′_E[ 1 : |E| ]
2.  parallel for u ← 1 to |V| do
3.      if M[ u ] = 1 then S_V[ u ] ← 0
4.  parallel for i ← 1 to |E| do
5.      u ← E[ i ].u, v ← E[ i ].v
6.      if M[ u ] = 1 or M[ v ] = 1 then
           S_V[ u ] ← 0, S_V[ v ] ← 0, S_E[ i ] ← 0
7.  S′_V ← Par-Prefix-Sum ( S_V, + ),
    S′_E ← Par-Prefix-Sum ( S_E, + )
8.  array U[ 1 : S′_V[ |V| ] ], F[ 1 : S′_E[ |E| ] ]
9.  parallel for u ← 1 to |V| do
10.     if S_V[ u ] = 1 then U[ S′_V[ u ] ] ← V[ u ]
11. parallel for i ← 1 to |E| do
12.     if S_E[ i ] = 1 then F[ S′_E[ i ] ] ← E[ i ]
13. parallel for i ← 1 to |F| do
14.     u ← F[ i ].u, v ← F[ i ].v
15.     F[ i ].u ← S′_V[ u ], F[ i ].v ← S′_V[ v ]
16. return ( U, F )
```

The prefix sums in line 7 perform $\Theta(|V| + |E|)$ work and have $\Theta(\log^2|V| + \log^2|E|)$ depth. The rest of the algorithm also perform $\Theta(|V| + |E|)$ work but in $\Theta(\log|V| + \log|E|)$ depth. Hence,

Work: $\Theta(|V| + |E|)$

Span: $\Theta(\log^2|V| + \log^2|E|)$

Randomized Maximal Independent Set (MIS)

```
Par-Randomized-MIS ( n, V, E, MIS )
 1.  while |V| > 0 do
 2.      array d[ 1 : |V| ], c[ 1 : |V| ] = { 0 },
                M[ 1 : |V| ] = { 0 }
 3.      parallel for i ← 1 to |E| do
 4.          if i = |E| or E[ i ].u ≠ E[ i + 1 ].u  then
                c[ E[ i ].u ] ← i
 5.      parallel for u ← 1 to |V| do
 6.          if u = 1 then d[ u ] ← c[ u ]
             else d[ u ] ← c[ u ] - c[ u - 1 ]
 7.          if d[ u ] = 0 then M[ u ] ← 1
 8.          else M[ u ] ← 1 ( with prob 1 / ( 2d[ u ] ) )
 9.      parallel for each ( u, v ) ∈ E do
10.          if M[ u ] = 1 and M[ v ] = 1 then
11.              if d[ u ] ≤ d[ v ] then M[ u ] ← 0
                else M[ v ] ← 0
12.      parallel for u ← 1 to |V| do
13.          if M[ u ] = 1 then MIS[ V[ u ] ] ← 1
14.      ( V, E ) ← Par-Compress ( V, E, M )
```

Let n = #vertices, and m = #edges initially.

Let us assume for the time being that at least a constant fraction of the edges are removed in each iteration of the *while* loop (we will prove this shortly). Let this fraction be f (< 1).

This implies that the *while* loop iterates $\Theta\left(\log_{1/(1-f)} m\right) = \Theta(\log m)$ times. (how?)

Each iteration performs $\Theta(|V| + |E|)$ work and has $\Theta(\log^2|V| + \log^2|E|)$ depth. Hence,

Work: $T_1(n,m) = \Theta\left((n+m)\sum_{i=0}^{k}(1-f)^i\right)$

$= \Theta(n+m)$

Span: $T_\infty(n,m) = \Theta((\log^2 n + \log^2 m)\log m)$

$= \Theta(\log^3 n)$

Parallelism: $\frac{T_1(n,m)}{T_\infty(n,m)} = \Theta\left(\frac{n+m}{\log^3 n}\right)$

Analysis of Randomized MIS

Let, $d(v)$ be the degree of vertex v, and $\Gamma(v)$ be its set of neighbors.

Good Vertex: A vertex v is *good* provided $|L(v)| \geq \frac{d(v)}{3}$, where,
$L(v) = \left\{ u \,\middle|\, \big(u \in \Gamma(v)\big) \wedge \big(d(u) \leq d(v)\big) \right\}$.

Bad Vertex: A vertex is *bad* if it is not good.

Good Edge: An edge (u, v) is *good* if at least one of u and v is good.

Bad Edge: An edge (u, v) is *bad* if both u and v are bad.

Analysis of Randomized MIS

Lemma 1: In some iteration of the *while* loop, let v be a good vertex with $d(v) > 0$, and let M be the set of vertices that got marked (in lines 7-8). Then

$$\Pr\{\,\Gamma(v) \cap M \neq \emptyset\,\} \geq 1 - e^{-1/6}.$$

Proof: We have, $\Pr\{\,\Gamma(v) \cap M \neq \emptyset\,\} = 1 - \Pr\{\,\Gamma(v) \cap M = \emptyset\,\}$

$$= 1 - \prod_{u\,\in \Gamma(v)} \Pr\{\,u \notin M\,\} \geq 1 - \prod_{u\,\in L(v)} \Pr\{\,u \notin M\,\}$$

$$= 1 - \prod_{u\,\in L(v)} \left(1 - \frac{1}{2d(u)}\right) \geq 1 - \prod_{u\,\in L(v)} \left(1 - \frac{1}{2d(v)}\right)$$

$$= 1 - \left(1 - \frac{1}{2d(v)}\right)^{|L(v)|} \geq 1 - \left(1 - \frac{1}{2d(v)}\right)^{d(v)/3}$$

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

Analysis of Randomized MIS

Lemma 2: In any iteration of the *while* loop, let M be the set of vertices that got marked (in lines 7-8), and let S be the set of vertices that got included in the MIS (in line 13). Then

$$\Pr\{\, v \in S \mid v \in M \,\} \geq \frac{1}{2}.$$

Proof: We have, $\Pr\{\, v \in S \mid v \in M \,\}$

$$\geq 1 - \Pr\{\, \exists u \in \Gamma(v) \text{ s.t. } (\, d(u) \geq d(v)\,) \wedge (\, u \in M\,)\}$$

$$\geq 1 - \sum_{\substack{u \in \Gamma(v) \\ d(u) \geq d(v)}} \frac{1}{2d(u)} \geq 1 - \sum_{\substack{u \in \Gamma(v) \\ d(u) \geq d(v)}} \frac{1}{2d(v)}$$

$$\geq 1 - \sum_{u \in \Gamma(v)} \frac{1}{2d(v)} = 1 - d(v) \times \frac{1}{2d(v)} = \frac{1}{2}$$

Analysis of Randomized MIS

Lemma 3: In any iteration of the *while* loop, let V_G be the set of good vertices, and let S be the vertex set that got included in the MIS. Then

$$\Pr\{\, v \in S \cup \Gamma(S) \mid v \in V_G \,\} \geq \frac{1}{2}\left(1 - e^{-1/6}\right).$$

Proof: We have, $\Pr\{\, v \in S \cup \Gamma(S) \mid v \in V_G \,\}$

$$\geq \Pr\{\, v \in \Gamma(S) \mid v \in V_G \,\} = \Pr\{\, \Gamma(v) \cap S \neq \phi \mid v \in V_G \,\}$$

$$= \Pr\{\, \Gamma(v) \cap S \neq \phi \mid \Gamma(v) \cap M \neq \phi, v \in V_G \,\} \\ \times \Pr\{\, \Gamma(v) \cap M \neq \phi \mid v \in V_G \,\}$$

$$\geq \Pr\{\, u \in S \mid u \in \Gamma(v) \cap M, v \in V_G \,\} \\ \times \Pr\{\, \Gamma(v) \cap M \neq \phi \mid v \in V_G \,\}$$

$$\geq \frac{1}{2}\left(1 - e^{-1/6}\right)$$

Analysis of Randomized MIS

Lemma 3: In any iteration of the *while* loop, let V_G be the set of good vertices, and let S be the vertex set that got included in the MIS. Then

$$\Pr\{\, v \in S \cup \Gamma(S) \mid v \in V_G \,\} \geq \frac{1}{2}\left(1 - e^{-1/6}\right).$$

Corollary 1: In any iteration of the *while* loop, a good vertex gets removed (in line 14) with probability at least $\frac{1}{2}\left(1 - e^{-1/6}\right)$.

Corollary 2: In any iteration of the *while* loop, a good edge gets removed (in line 14) with probability at least $\frac{1}{2}\left(1 - e^{-1/6}\right)$.

Analysis of Randomized MIS

Lemma 4: In any iteration of the *while* loop, let E and E_G be the sets of all edges and good edges, respectively. Then $|\mathrm{E_G}| \geq |E|/2$.

Proof: For each edge $(u, v) \in E$, direct (u, v) from u to v if $d(u) \leq d(v)$, and v to u otherwise.

For every vertex v in the resulting digraph let $d_i(v)$ and $d_o(v)$ denote its in-degree and out-degree, respectively.

Let V_G and V_B be the set of good and bad vertices, respectively.

Then for each $v \in V_B$, $d_o(v) - d_i(v) \geq \frac{d(v)}{3}$.

Let m_{BB}, m_{BG}, m_{GB} and m_{GG} be the #edges directed from V_B to V_B, from V_B to V_G, from V_G to V_B, and from V_G to V_G, respectively.

Analysis of Randomized MIS

Lemma 4: In any iteration of the *while* loop, let E and E_G be the sets of all edges and good edges, respectively. Then $|E_G| \geq |E|/2$.

Proof (continued): We have,

$$2m_{BB} + m_{BG} + m_{GB}$$

$$= \sum_{v \in V_B} d(v) \leq 3 \sum_{v \in V_B} \big(d_o(v) - d_i(v)\big) = 3 \sum_{v \in V_G} \big(d_i(v) - d_o(v)\big)$$

$$= 3\big((m_{BG} + m_{GG}) - (m_{GB} + m_{GG})\big) = 3(m_{BG} - m_{GB})$$

$$\leq 3(m_{BG} + m_{GB})$$

Thus $2m_{BB} + m_{BG} + m_{GB} \leq 3(m_{BG} + m_{GB})$

$$\Rightarrow m_{BB} \leq m_{BG} + m_{GB} \Rightarrow m_{BB} \leq m_{BG} + m_{GB} + m_{GG}$$

$$\Rightarrow (m_{BG} + m_{GB} + m_{GG}) + \; m_{BB} \leq 2(m_{BG} + m_{GB} + m_{GG})$$

$$\Rightarrow |E| \leq 2|E_G|$$

Analysis of Randomized MIS

Lemma 5: In any iteration of the *while* loop, let E be the set of all edges. Then the expected number of edges removed (in line 14) during this iteration is at least $\frac{1}{4}\left(1 - e^{-1/6}\right)|E|$.

Proof: Follows from Lemma 4 and Corollary 2.