Network Computing and Efficient Algorithms

Maximal Independent Set

Xiang-Yang Li and Xiaohua Xu

School of Computer Science and Technology
University of Science and Technology of China (USTC)

September 1, 2021

Outline

Maximal Independent Set

Definition 7.1 (Independent Set)

Given an undirected Graph $G = (V, E)$ an independent set is a subset of nodes $U \subseteq V$,such that no two nodes in U are adjacent. An independent set is a **maximal** if no node can be added without violating independence. An independent set of **maximum** cardinality is called maximum.

Maximal Independent Set

Definition 7.1 (Independent Set)

Given an undirected Graph $G = (V, E)$ an independent set is a subset of nodes $U \subseteq V$,such that no two nodes in U are adjacent. An independent set is a **maximal** if no node can be added without violating independence. An independent set of **maximum** cardinality is called maximum.

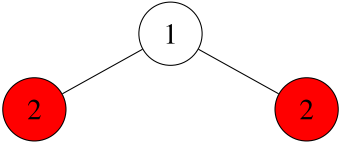


Figure 7.2: Example graph with 1) a maximal independent set (MIS) and 2) a maxinum independent set (MaxIS).

Remarks

- Computing a maximum independent set (MaxIS) is equivalent to computing maximum clique on the complementary graph.
  - Both problems are NP-hard.
  - Not approximable within $n^{0.5-\varepsilon}$ within polynomial time.
- MIS and MaxIS can be quite different.
- Computing a MIS sequentially is trivial...
  - How to compute a MIS in a distributed way?

Slow MIS

Require: Node IDs
Every node *v* executes the following code:

ALGORITHM 7.3: SLOW MIS()

1: **if** all neighbours of *v* with lager identifiers have decided not to join the MIS **then**
2: *v* decides to join the MIS

Slow MIS

Require: Node IDs
Every node v executes the following code:

ALGORITHM 7.3: SLOW MIS()

1: **if** all neighbours of v with lager identifiers have decided not to join the MIS **then**
2: v decides to join the MIS

Theorem 7.4 (Analysis of Algorithm 7.3).

Algorithm 7.3 features a time complexity of $O(n)$ and a message complexity of $O(m)$

Slow MIS

Require: Node IDs
Every node v executes the following code:

Algorithm 7.3: Slow MIS()

1: **if** all neighbours of v with lager identifiers have decided not to join the MIS **then**
2: v decides to join the MIS

Theorem 7.4 (Analysis of Algorithm 7.3).

Algorithm 7.3 features a time complexity of $O(n)$ and a message complexity of $O(m)$

Not better than the sequential algorithm in the worst case!

Using Vertex Coloring

- **Basic idea**
  - Each color class is an independent set
  - Not necessarily a MIS

Using Vertex Coloring

- **Basic idea**
  - Each color class is an independent set
  - Not necessarily a MIS
- **MIS algorithm using vertex coloring:**
  - In the first round all nodes of the first color join the MIS and notify their neighbors.
  - Then, all nodes of the second color which do not have a neighbor that is already in the MIS join the MIS and inform their neighbors.
  - This process is repeated for all colors.

Using Vertex Coloring

- **Basic idea**
  - Each color class is an independent set
  - Not necessarily a MIS
- **MIS algorithm using vertex coloring:**
  - In the first round all nodes of the first color join the MIS and notify their neighbors.
  - Then, all nodes of the second color which do not have a neighbor that is already in the MIS join the MIS and inform their neighbors.
  - This process is repeated for all colors.

Corollary 7.5

Given a coloring algorithm that runs in time T and needs C colors we can construct a MIS in time $T + C$

Fast MIS

The algorithm operates in synchronous rounds, grouped into phases.
A single phase is as follows:

ALGORITHM 7.6: FAST MIS()

1: Each node v marks itself with probability $\frac{1}{2d(v)}$, where $d(v)$ is the current degree of v.
2: If no higher degree neighbour of v is also marked, node v joins the MIS. If a higher degree neighbor if v is marked, node v unmarks itself again (If the neighbors have the same degree, tries a broken arbitrarily, *e.g.*, by identifier).
3: Delete all nodes that joined the MIS and their neighbors, as they cannot join the MIS anymore.

Fast MIS

The algorithm operates in synchronous rounds, grouped into phases.
A single phase is as follows:

ALGORITHM 7.6: FAST MIS()

1: Each node v marks itself with probability $\frac{1}{2d(v)}$, where $d(v)$ is the current degree of v.
2: If no higher degree neighbour of v is also marked, node v joins the MIS. If a higher degree neighbor if v is marked, node v unmarks itself again (If the neighbors have the same degree, tries a broken arbitrarily, *e.g.*, by identifier).
3: Delete all nodes that joined the MIS and their neighbors, as they cannot join the MIS anymore.

Theorem 7.11 (Analysis of Algorithm 7.6)

Algorithm 7.6 terminates in expected time $O(\log n)$

Fast MIS terminates in expeded time $O(\log n)$

$d(v)$: node degree of v
$N(v)$: set of neighbors of v
$H(v)$: set of neighbors of v with higher degree …
M: set of nodes marked in Step 1.

Fast MIS terminates in expeded time $O(\log n)$

Lemma 7.1(Joining MIS).

A node v joins the MIS in Step 2 with probability $P \geq \frac{1}{4d(v)}$

Fast MIS terminates in expeded time $O(\log n)$

Lemma 7.1(Joining MIS).

A node v joins the MIS in Step 2 with probability $P \geq \frac{1}{4d(v)}$

$$\begin{aligned} P[v \notin MIS|v \in M] &= P[there\ is\ a\ node\ w \in H(v), w \in M|v \in M] \\ &= P[there\ is\ a\ node\ w \in H(v), w \in M] \\ &\leq \sum_{w \in H(v)} P[w \in M] = \sum_{w \in H(v)} \frac{1}{2d(w)} \\ &\leq \sum_{w \in H(v)} \frac{1}{2d(v)} \leq \frac{d(v)}{2d(v)} = \frac{1}{2} \end{aligned}$$

Then

$$P[v \in MIS] = P[v \in MIS|v \in M] \cdot P[v \in M] \geq \frac{1}{2} \cdot \frac{1}{2d(v)}$$

Fast MIS terminates in expeded time $O(\log n)$

Lemma 7.8(Good Nodes).

A node v is called good if

$$\sum_{w \in N(v)} \frac{1}{2d(w)} \geq \frac{1}{6}$$

where $N(v)$ is the set of neighbors of v. Otherwise we call v a bad node. **A good node will be removed in Step 3 with probability $p \geq \frac{1}{36}$.**

Fast MIS terminates in expeded time $O(\log n)$

Lemma 7.8(Good Nodes).

A node v is called good if

$$\sum_{w \in N(v)} \frac{1}{2d(w)} \geq \frac{1}{6}$$

where $N(v)$ is the set of neighbors of v. Otherwise we call v a bad node. **A good node will be removed in Step 3 with probability $p \geq \frac{1}{36}$.**

Lemma 7.9(Good Edges).

An edge $e = (u, v)$ is called bad if both u and v are bad; else the edge is called good. The following holds:**At any time at least half of the edges are good**

Fast MIS v2

The algorithm operates in synchronous rounds, grouped into phases.
A single phase is as follows:

ALGORITHM 7.12: FAST MIS V2()

1: Each node v chooses a random value $r(v) \in [0, 1]$ and send it to its neighbors.
2: if $r(v) < r(w)$ for all neighbors $w \in N(v)$, node v enters the MIS and informs its neighbors.
3: if v or a neighbor of v entered the MIS, v terminates (v and all edges adjacent to v are removed from the graph), otherwise v enters the nest phase.

Corollary 7.18(Running Time of Algorithm 7.12)

Algorithm 7.12 terminates $w.h.p.$ in $O \log n$ time.

Application for Matching

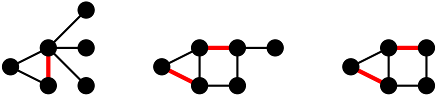
Definition 7.19 (Matching)

Given a graph $G = (V, E)$ a matching is a subset of edges $M \subseteq E$, such that no two edges in M are adjacent(*i.e.*, where no node is adjacent to two edges in the matching). A matching is **maximal** if no edge can be added without violating the above constraint. A matching is **maximum** cardinality is called maximum. A matching is called **perfect** if each node is adjacent to an edge in the matching

Application for Matching

Definition 7.19 (Matching)

Given a graph $G = (V, E)$ a matching is a subset of edges $M \subseteq E$, such that no two edges in M are adjacent(*i.e.*, where no node is adjacent to two edges in the matching). A matching is **maximal** if no edge can be added without violating the above constraint. A matching is **maximum** cardinality is called maximum. A matching is called **perfect** if each node is adjacent to an edge in the matching

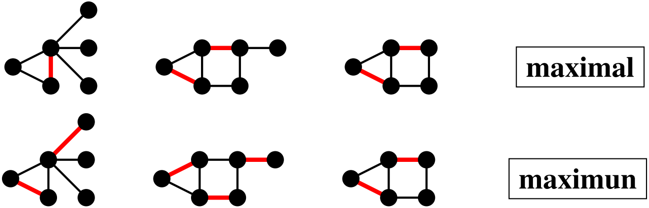


maximal

Application for Matching

Definition 7.19 (Matching)

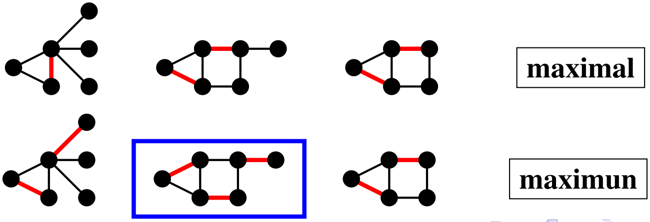
Given a graph $G = (V, E)$ a matching is a subset of edges $M \subseteq E$, such that no two edges in M are adjacent(*i.e.*, where no node is adjacent to two edges in the matching). A matching is **maximal** if no edge can be added without violating the above constraint. A matching is **maximum** cardinality is called maximum. A matching is called **perfect** if each node is adjacent to an edge in the matching



Application for Matching

Definition 7.19 (Matching)

Given a graph $G = (V, E)$ a matching is a subset of edges $M \subseteq E$, such that no two edges in M are adjacent(*i.e.*, where no node is adjacent to two edges in the matching). A matching is **maximal** if no edge can be added without violating the above constraint. A matching is **maximum** cardinality is called maximum. A matching is called **perfect** if each node is adjacent to an edge in the matching



Application for Matching

- In constrast to MaxIS, a maximum matching can be found in polynomial time and also easi to approximate, since any maximal matching is a 2-approximation.
- An independent set algorithm is also a matching algorithm: Let $G = (V, E)$ be the graph for which we want to construct the matching. The so-called line graph G' is defined as follows: for every edge in G there is a node in G'; two nodes in G' are connected by an edge if their respective edges in G are adjacent. A (maximal) independent set in the line graph G' is a (maximal) matching in the original graph G, and vice versa. Using Algorithm 7.12 directly produces a $O(\log n)$ bound for maximal matching.

Application for Coloring

ALGORITHM 7.20 GENERAL GRAPH COLORING()

1: Given a graph $G = (V, E)$ we virtually build a graph $G' = (V', E')$ as follows:
2: Every node $v \in V$ clones itself $d(v) + 1$ times $(v_0, \ldots, v_{d(v)} \in V')$,$d(v)$ being the degree of v in G.
3: The edge set E' of G' is as follows:
4: First all clones are in a clique: $(v_i, v_j) \in E'$, for all $v \in V$ and all $0 \leq i \leq j \leq d(v)$.
5: Second all i^{th} clones of neighbors in the original graph G are connected:$(u_i, v_i) \in E'$, for all $(u, v) \in E$ and all $0 \leq i \leq min(d(u), d(v))$.
6: Now we simply run (simulate) the fast MIS algorithm 7.12 on G'.
7: if node v_i is in the MIS in G', then node v gets color i.

Application for Coloring

Theorem 7.21(Analysis of Algorithm 7.20).

Algorithm 7.20 $(\Delta+1)$-colors an arbitrary graph in $O(\log n)$ time, with high probability, Δ being the largest degree in the graph.

Application for Coloring

Theorem 7.21(Analysis of Algorithm 7.20).

Algorithm 7.20 $(\Delta + 1)$-colors an arbitrary graph in $O(\log n)$ time, with high probability, Δ being the largest degree in the graph.

Proof: Thanks to the clique among the clones at most one clone is in the MIS. And because of the $d(v)+1$ clones of node v every node will get a free color! The running time remains logarithmic since G' has $O(n^2)$n nodes and the exponent becomes a constant factor when applying the logarithm.

Application for Coloring

Theorem 7.21(Analysis of Algorithm 7.20).

Algorithm 7.20 $(\Delta+1)$-colors an arbitrary graph in $O(\log n)$ time, with high probability, Δ being the largest degree in the graph.

Proof: Thanks to the clique among the clones at most one clone is in the MIS. And because of the $d(v)+1$ clones of node v every node will get a free color! The running time remains logarithmic since G' has $O(n^2)$n nodes and the exponent becomes a constant factor when applying the logarithm.

- Together with Corollary 7.5 we get quite close ties between $(\Delta+1)$-coloring and the MIS problem.
- Computing a MIS also solves another graph problem on graphs of bounded independence.

Application for Dominating Sets

Definition 7.22(Bounded Independence).

$G = (V, E)$ is of bounded independence, if for every node $v \in V$ the largest independent set in the neighborhood $N(v)$ is bounded by a constant.

Definition 7.22((Minimum) Dominating Sets).

A dominating set is a subset of the nodes such that each node is the set or adjacent to a node in the set.
A minimum dominating set is a dominating set containing the least possible number of nodes.

Application for Dominating Sets

Definition 7.22(Bounded Independence).

$G = (V,E)$ is of bounded independence, if for every node $v \in V$ the largest independent set in the neighborhood $N(v)$ is bounded by a constant.

Definition 7.22((Minimum) Dominating Sets).

A dominating set is a subset of the nodes such that each node is the set or adjacent to a node in the set.
A minimum dominating set is a dominating set containing the least possible number of nodes.

- In general, finding a dominating set less than a factor $\log n$ larger than a minimum dominating set is NP-hard.
- Any MIS is a dominating set: if a node was not covered, it could join the independent set.
- In general a MIS and a minimun dominating set have not much in common (think of a star). For graphs of bounded Independence, this is different.

Application for Dominating Sets

Corollary 7.24.

On graphs of bounded independence, a constant-factor approximation to a minimum dominating set can be found in the time $O(\log n)$ *w.h.p.*

Application for Dominating Sets

Corollary 7.24.

On graphs of bounded independence, a constant-factor approximation to a minimum dominating set can be found in the time $O(\log n)$ $w.h.p.$

Proof: Denote by M a minimum dominating set and by I a MIS. Since M is a dominating set, each node from I is in M or adjacent to a node in M. Since the graph is of bounded independence, no node in M is adjacent to more than constantly many nodes from I. Thus, $|I| \in O(|M|)$. Therefore, we can compute a MIS woth Algorithm 7.12 and output it as the dominating set, which takes $O(\log n)$ rounds $w.h.p.$

References

- MIS results that depend on the degree of the graph:
  - Leonid Barenboim and Michael Elkin. Distributed (delta+1)-coloring in linear (in delta) time. In STOC, 2009.
  - Fabian Kuhn. Weak graph colorings: distributed algorithms and applications. In SPAA, 2009.
- The strongest MIS time complexity lower bounds so far:
  - F. Kuhn, T. Moscibroda, and R. Wattenhofer. What Cannot Be Computed Locally! In PODC, 2004.
- Recent research focused on improving the O(log n) time complexity for special graph classes:
  - Christoph Lenzen and Roger Wattenhofer. MIS on trees. In PODC, pages 41-48, 2011.
  - Johannes Schneider and Roger Wattenhofer. A Log-Star Distributed Maximal Independent Set Algorithm for Growth-Bounded Graphs. In PODC, 2008.