Dynamic Maximal Independent Sets

Presentation 1

Thilo L. Fischer

May 19, 2020

Outline

Introduction

Introduction

$$G = (V, E)$$
$$n := |V|$$
$$m := |E|$$
$$\Delta := max\{deg(v)\}$$

Independent Set (IS):

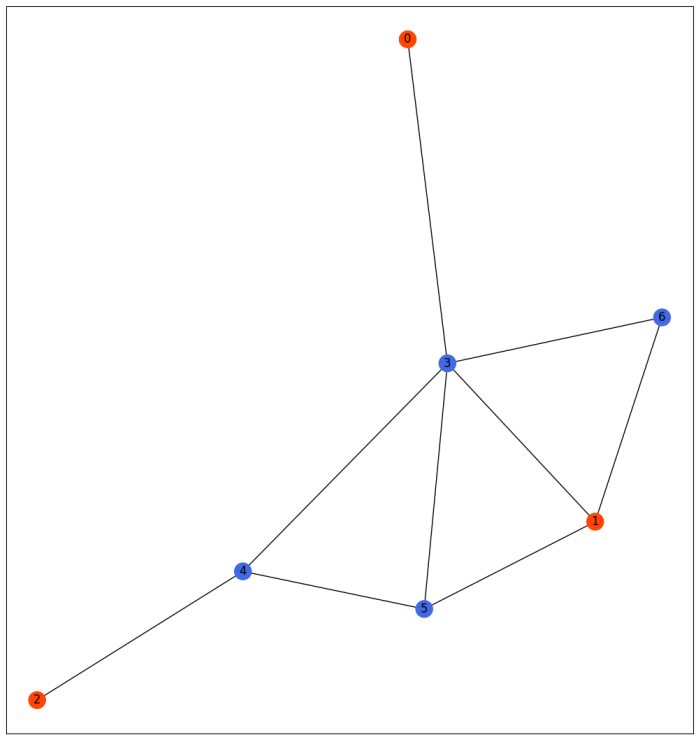
$$\mathcal{M} \subset V : \ \forall u, v \in \mathcal{M} : \ \{u, v\} \notin E$$

$\mathcal{M}$ is **maximal** if (MIS):

$$\nexists \ IS \ \mathcal{M}' : \mathcal{M} \subsetneq \mathcal{M}'$$

Example MIS

- Orange: in the MIS
- Blue: not in MIS



Intuitive Approach

Intuitive Approach $\mathcal{A}_T$

- Iterate over all vertices
- Check if vertex can be in MIS
- Complexity: $\mathcal{O}(m)$

Intuitive Approach $\mathcal{A}_T$

- Iterate over all vertices
- Check if vertex can be in MIS
- Complexity: $\mathcal{O}(m)$

- Recompute after every update

Types of updates

- Node removal
- Node insertion
- Edge removal
- Edge insertion

- 1 node might leave $\mathcal{M}$
- up to Δ might enter $\mathcal{M}$

Simple Dynamic Algorithm

Simple Dynamic Algorithm $\mathcal{A}_S$

- Every node has counter of neighbors in $\mathcal{M}$
- If counter $= 0$: node can be added

Simple Dynamic Algorithm $\mathcal{A}_S$

- Every node has counter of neighbors in $\mathcal{M}$
- If counter $= 0$: node can be added
- Every node joining/leaving $\mathcal{M}$ has to inform $\mathcal{O}(\Delta)$ neighbors
- amortized time per update: $\mathcal{O}(\Delta)$

Simple Dynamic Algorithm $\mathcal{A}_S$

- Every node has counter of neighbors in $\mathcal{M}$
- If counter $= 0$: node can be added
- Every node joining/leaving $\mathcal{M}$ has to inform $\mathcal{O}(\Delta)$ neighbors
- amortized time per update: $\mathcal{O}(\Delta)$
- Why? $\mathcal{O}(\Delta)$ insertions costing $\mathcal{O}(\Delta)$ each!

Simple Dynamic Algorithm $\mathcal{A}_S$

- Every node has counter of neighbors in $\mathcal{M}$
- If counter $= 0$: node can be added
- Every node joining/leaving $\mathcal{M}$ has to inform $\mathcal{O}(\Delta)$ neighbors
- amortized time per update: $\mathcal{O}(\Delta)$
- Why? $\mathcal{O}(\Delta)$ insertions costing $\mathcal{O}(\Delta)$ each!
  $\rightarrow$ Assign cost of next insertion to removal

Improved Dynamic Algorithm

Improved Dynamic Algorithm

- Complexity of $\mathcal{A}_S$ is determined by high degree nodes

Improved Dynamic Algorithm

- Complexity of $\mathcal{A}_S$ is determined by high degree nodes
- Idea: differentiate between **heavy** and **light** vertices
- v is **light** if: $deg(v) < \Delta_C$

Improved Dynamic Algorithm

- Complexity of $\mathcal{A}_S$ is determined by high degree nodes
- Idea: differentiate between **heavy** and **light** vertices
- v is **light** if: $deg(v) < \Delta_C$

1. $\mathcal{A}_S$ for light vertices
2. $\mathcal{A}_T$ for heavy vertices
    - respect the IS on the light nodes!

Improved Dynamic Algorithm

- $\Delta_C := m^{2/3}$

Improved Dynamic Algorithm

- $\Delta_C := m^{2/3}$
  $\rightarrow$ New phase if $m \geq 2m_c \vee m \leq m_c/2$

Improved Dynamic Algorithm

- $\Delta_C := m^{2/3}$
  $\rightarrow$ New phase if $m \geq 2m_c \vee m \leq m_c/2$

- $\mathcal{A}_S$ takes $\mathcal{O}(m^{2/3})$
- $\mathcal{A}_T$ takes $\mathcal{O}(m_H) = \mathcal{O}((\frac{m}{\Delta_C})^2) = \mathcal{O}(m^{2/3})$

Improved Dynamic Algorithm

- $\Delta_C := m^{2/3}$
  $\rightarrow$ New phase if $m \geq 2m_c \vee m \leq m_c/2$

- $\mathcal{A}_S$ takes $\mathcal{O}(m^{2/3})$
- $\mathcal{A}_T$ takes $\mathcal{O}(m_H) = \mathcal{O}((\frac{m}{\Delta_C})^2) = \mathcal{O}(m^{2/3})$

- overall amortized complexity: $\mathcal{O}(min\{\Delta, m^{2/3}\})$
  $\rightarrow$ all nodes could be light

Implicit MIS Algorithm

Relaxed Model

- The previous algorithms:
  - kept explicit copy of MIS
  - analyzed for amortized complexity

Relaxed Model

- The previous algorithms:
  - kept explicit copy of MIS
  - analyzed for amortized complexity
- Relax requirement to keep explicit copy of MIS

Implicit MIS $\mathcal{A}_I$

- Only maintain an IS not an MIS
- Only process removals of nodes

Implicit MIS $\mathcal{A}_I$

- Only maintain an IS not an MIS
- Only process removals of nodes
- Check if $v \in \mathcal{M}$ when asked
- Worst case complexity $\mathcal{O}(min\{\Delta, \sqrt{m}\})$

Realisation of $\mathcal{A}_I$

- Again **light** and **heavy** nodes
- $\Delta_C := \sqrt{m}$

Realisation of $\mathcal{A}_I$

- Again **light** and **heavy** nodes
- $\Delta_C := \sqrt{m}$
- Maintain count for heavy nodes only
- Compute status for light nodes when queried

Conclusion

- Code and slides at:
  github.com/thilofischer/dynamic_mis