Proof methods in structural graph theory

Tara Abrishami

PACM Graduate Student Seminar, February 2021

Introduction

Introduction

An **independent set** in a graph G is a set $I \subseteq V(G)$ such that no edge in $E(G)$ has both endpoints in I.

Introduction

An **independent set** in a graph G is a set $I \subseteq V(G)$ such that no edge in $E(G)$ has both endpoints in I.

Maximum independent set (MIS): Can we find an algorithm that, given a graph G, finds a maximum independent set of G in polynomial time?

Introduction

An **independent set** in a graph G is a set $I \subseteq V(G)$ such that no edge in $E(G)$ has both endpoints in I.

Maximum independent set (MIS): Can we find an algorithm that, given a graph G, finds a maximum independent set of G in polynomial time?

Answer: NO! MIS is **NP-hard** (Karp, 1972)

Introduction

An **independent set** in a graph G is a set $I \subseteq V(G)$ such that no edge in $E(G)$ has both endpoints in I.

Maximum independent set (MIS): Can we find an algorithm that, given a graph G, finds a maximum independent set of G in polynomial time?

Answer: NO! MIS is **NP-hard** (Karp, 1972)

Maximum independent set (MIS): Can we find an algorithm that, given a graph G *with some constraints on* G, finds a maximum independent set of G in polynomial time?

Introduction

An **induced subgraph** of G is a subgraph of G formed by vertex deletions

Important induced subgraphs

- Cycle on k vertices C_k
- Path on k vertices P_k

If H is a graph, then **H-free graphs** are graphs with no induced H.

If $\mathcal{H}$ is a set of graphs, then **$\mathcal{H}$-free graphs** are graphs with no induced H for every $H \in \mathcal{H}$.

Tree decompositions

A **tree decomposition** of a graph G is a partition of G into a "tree-like" structure. The **treewidth** of G measures how "close" to a tree G is.

Tree decompositions

A **tree decomposition** of a graph G is a partition of G into a "tree-like" structure. The **treewidth** of G measures how "close" to a tree G is.

Tree decompositions are useful for algorithms because of **dynamic programming**, a method which breaks a problem into pieces and then combines the results.

Separations

A **separation** of a graph G is a triple (A, C, B), such that $A \cup C \cup B = V(G)$, A, B, and C are disjoint, and A is anticomplete to B.

Separations

A **separation** of a graph G is a triple (A, C, B), such that $A \cup C \cup B = V(G)$, A, B, and C are disjoint, and A is anticomplete to B.

If G has a **balanced separation**, then algorithmic problems can be solved quickly using recursion.

Separations

If (A, C, B) is a separation of G, assume that B is very large.

Separations

If (A, C, B) is a separation of G, assume that B is very large.

Two separations (A_1, C_1, B_1) and (A_2, C_2, B_2) are **non-crossing** if A_1 and A_2 are disjoint and anticomplete.

Central bag

Given a collection $\mathcal{S}$ of non-crossing separations, the **central bag for $\mathcal{S}$**, denoted β, is defined as follows:

$$\beta = \bigcap_{(A,C,B) \in \mathcal{S}} (B \cup C)$$

Forcers

Usually, more restrictions on a graph's structure = better information and better algorithms

Example: graphs with no odd cycle

Forcers

Usually, more restrictions on a graph's structure = better information and better algorithms

Example: graphs with no odd cycle

Want: central bags to be structurally restricted

Forcers

Usually, more restrictions on a graph's structure = better information and better algorithms

Example: graphs with no odd cycle

Want: central bags to be structurally restricted

Tool: **Forcers!**

Forcers

An graph F is a **forcer** for G if for every induced subgraph H of G isomorphic to F, there exists a separation (A, C, B) of G such that $F \cap A \neq \emptyset$.

Approach outline

Approach outline:

1. List some forcers $\mathcal{F}$ for a graph G
2. List all the separations $\mathcal{S}$ that correspond to forcers in $\mathcal{F}$
3. Partition the separations $\mathcal{S}$ into non-crossing collections $\mathcal{S}_1, \ldots, \mathcal{S}_k$
4. Take central bags $\beta_1, \ldots, \beta_k$ for the collections $\mathcal{S}_1, \ldots, \mathcal{S}_k$
5. β_k contains no forcers in , so β_k has a nice structure
6. Use the nice structure of β_k to draw conclusions about the structure of G

The End

Questions?