Subgraph isomorphism problems

Tamás Takács

Supervisor: Péter Madarasi

June 4, 2026

Subgraph isomorphism

Problem

For two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, we want to find an induced subgraph of G_2 that is isomorphic to G_1.
In the weighted case, for a given weight function $w : V_1 \times V_2 \to \mathbb{R}$, we want to find an embedding of G_1 into G_2 with maximum weight.

Clique problem

Goal: Find a maximum-weight embedding of G_1 into G_2.

Clique problem

Goal: Find a maximum-weight embedding of G_1 into G_2.

Definition

We call the graph $G = (V_1 \times V_2, E)$ the product of G_1 and G_2, where there is an edge between (u_1, u_2) and (v_1, v_2) if and only if $u_1v_1 \in E_1$ and $u_2v_2 \in E_2$, or $u_1v_1 \notin E_1$ and $u_2v_2 \notin E_2$.

Clique problem

Goal: Find a maximum-weight embedding of G_1 into G_2.

Definition

We call the graph $G = (V_1 \times V_2, E)$ the product of G_1 and G_2, where there is an edge between (u_1, u_2) and (v_1, v_2) if and only if $u_1v_1 \in E_1$ and $u_2v_2 \in E_2$, or $u_1v_1 \notin E_1$ and $u_2v_2 \notin E_2$.

If $|V_1| = k$, then G is k-partite, and the k-cliques in G correspond to the embeddings of G_1 into G_2.
Finding a maximum-weight embedding is therefore equivalent to finding a maximum-weight k-clique in the product graph.

Protein docking

Goal: Predict the position of a small molecule when it is bound to a protein.

Protein docking

Goal: Predict the position of a small molecule when it is bound to a protein.

- Proteins can be represented by protein graphs.

Protein docking

Goal: Predict the position of a small molecule when it is bound to a protein.

- Proteins can be represented by protein graphs.
- To predict the position of the molecule, we need to find the embeddings of the small protein graph into the bigger protein graph with the best weights.

Protein docking

Goal: Predict the position of a small molecule when it is bound to a protein.

- Proteins can be represented by protein graphs.
- To predict the position of the molecule, we need to find the embeddings of the small protein graph into the bigger protein graph with the best weights.
- We do this by finding the 100 greatest weight k-cliques in the product graph.

Kernelization for the clique problem

$G = (V, E)$, color classes $C_1, \ldots, C_k$

Kernelization for the clique problem

$G = (V, E)$, color classes $C_1, \ldots, C_k$

- If there exists C_i s.t. $N(v) \cap C_i = \emptyset \Rightarrow$ we can delete v.

Kernelization for the clique problem

$G = (V, E)$, color classes $C_1, \ldots, C_k$

- If there exists C_i s.t. $N(v) \cap C_i = \emptyset \Rightarrow$ we can delete v.
- No edge between $N(v) \cap C_i$ and $N(v) \cap C_j$ for some $i, j \Rightarrow$ we can delete v.

Kernelization for the clique problem

$G = (V, E)$, color classes $C_1, \ldots, C_k$

- If there exists C_i s.t. $N(v) \cap C_i = \emptyset \Rightarrow$ we can delete v.
- No edge between $N(v) \cap C_i$ and $N(v) \cap C_j$ for some $i, j \Rightarrow$ we can delete v.
- If for an edge uv there is no edge between $C_i \cap N(u) \cap N(v)$ and $C_j \cap N(u) \cap N(v) \Rightarrow$ we can delete the edge uv.

Kernelization for the clique problem

$G = (V, E)$, color classes $C_1, \ldots, C_k$

- If there exists C_i s.t. $N(v) \cap C_i = \emptyset \Rightarrow$ we can delete v.
- No edge between $N(v) \cap C_i$ and $N(v) \cap C_j$ for some $i, j \Rightarrow$ we can delete v.
- If for an edge uv there is no edge between $C_i \cap N(u) \cap N(v)$ and $C_j \cap N(u) \cap N(v) \Rightarrow$ we can delete the edge uv.
- Dominance relations
- Edge-to-Node transformation

Our work

- We became familiar with the insidrug codebase used to solve protein docking and added a naive weighted clique search.

Our work

- We became familiar with the insidrug codebase used to solve protein docking and added a naive weighted clique search.
- We created test instances to test the runtime of our algorithms.

Our work

- We became familiar with the insidrug codebase used to solve protein docking and added a naive weighted clique search.
- We created test instances to test the runtime of our algorithms.
- We started adding heuristics to our algorithm.

Backtracking step

1. Greedily color the vertices to obtain color classes $C_1, \ldots, C_k$.
2. Sort the color classes by non-increasing size.
3. For $i = 1, \ldots, k$ and every $v \in C_i$:

$$UB_1(v) = \sum_{j=1}^{i} \max\{w(u) : u \in C_j \cap N(v)\},$$

$$UB_2(v) = \left|\{j \in [i] : C_j \cap N(v) \neq \emptyset\}\right|.$$

4. Use UB_1 and UB_2 to prune the search tree.

Future plans

We plan to

Future plans

We plan to

- Add more heuristics to our weighted clique search algorithm to solve protein docking quicker,

Future plans

We plan to

- Add more heuristics to our weighted clique search algorithm to solve protein docking quicker,
- Find a way to avoid infeasible embeddings,

Future plans

We plan to

- Add more heuristics to our weighted clique search algorithm to solve protein docking quicker,
- Find a way to avoid infeasible embeddings,
- Apply the kernelization methods directly to the subgraph problem.

Usage of AI tools

Throughout the project, no AI tools were used.