Searching and generating sparse (sub)graphs

Bence Deák

Applied Mathematics MSc

Thursday 9th January, 2025

Course: Math project I.
Supervisor: Péter Madarasi

(k, l)-sparsity

Definition

A graph $G = (V, E)$ is (k, l)-sparse if $i(X) \leq \max\{0, k|X| - l\}$ for all $X \subseteq V$. It is (k, l)-tight if $|E| = k|V| - l$ also holds.

Example (Nash-Williams)

G can be covered by k forests $\Leftrightarrow$ G is (k, k)-sparse.

Example (Laman)

G is minimally rigid on the plane $\Leftrightarrow$ G is $(2, 3)$-tight.

The pebble game algorithm

Problem

Find the maximum size/weight sparse subgraph of a graph.

Solution

- *Try to insert the edges greedily*
- *Maintain an orientation of the subgraph*
- *Reverse some paths in each step*

Time complexity: $O(nm)$, *or* $O(n^2)$ *if we maintain the components.*

Data structure for a pebble game heuristic

Problem

Perform the following operations on a graph G with a weight function w on its vertices:

- *Find the edge uv that maximizes $w(u) + w(v)$*
- *Increase/decrease $w(u)$ by 1*

Solution

- *Naive (1 priority queue): amortized $O(1)$ query and $O(\Delta)$ update*
- **Sqrt decomposition** *($O(\sqrt{m})$ priority queues): amortized $O(\sqrt{m})$ query and update*

Finding a maximum weight sparse subgraph in $O(n^2)$ time

Problem

Maintain the components during the pebble game algorithm to achieve an $O(n^2)$ total running time.

Solution

- *$B_{u,v}$ indicates whether u and v are in a common component*
- *Update B accordingly upon merging*

The original analysis is flawed. Instead, we bound the number of 1 to 1 (*redundant*) modifications in B:

- When merging $C_1, \ldots, C_t$, at most $4t^2$ *redundant* modifications
- $O(n)$ components arise, so $O(n^2)$ *redundant* modifications in total

Faster sparse graph generation

Problem

Generate all non-isomorphic (k, l)-sparse graphs up to a given size.

Solution

- *Recursively add vertices*
- *For a new vertex s, check sparsity for all neighbourhoods*
- *Use canonization to avoid duplicates*

Possible approaches for sparsity checking:

- *Naive: $O(4^n)$ time*
- *Pebble game: $O(2^n \cdot n^2)$ time (high constant, not general)*

Improvement: **precalculation with DP**, *answer all checks in $O(2^n \cdot n)$.*

Continuation

Plans for the next semester:

- Implementation of the sparse graph filtering subroutine
- Improvements in the priority-queue-like data structure
- Generalize the component-based pebble game algorithm
- Heuristic improvements in the weighted optimization problem