Karp and Widger Fast MIS

Bogdan Tolstik

January 21, 2025

Main Algorithm Structure

The main algorithm structure is described as follows:

Algorithm 1 Main Algorithm Structure

$I \leftarrow \emptyset$ // Initialize empty solution
$H \leftarrow V$ // All vertices active
while $H \neq \emptyset$ **do**
  $K \leftarrow \text{HEAVYFIND}(H)$ // Find set with frequent vertices
  $T \leftarrow \text{SCOREFIND}(K)$ // Select promising subset
  $S \leftarrow \text{INDFIND}(T)$ // Ensure independence
  $I \leftarrow I \cup S$ // Add to solution
  $H \leftarrow H - (S \cup N_H(S))$ // Remove used vertices
end while
return I

INDFIND Algorithm

The `INDFIND` algorithm ensures independence within the selected set T:

Algorithm 2 INDFIND(T)

For each pair $\{u, w\} \subseteq T$:
if $(u, w) \in E(T)$ **then**
  Remove random vertex
end if
return remaining vertices

Comments:

- There are 3 rounds in this algorithm:
  1. In the first round, each vertex exchanges its ID with its neighbors.

2. In the second round, the leader (the vertex with the highest ID among pairs) decides who will be removed and sends the bit.
3. In the third round if we are not removed we become inactive and we notify our neighbours that they became inactive too, or that everything is fine and we both are active. We also add flag that we are in maximal independent set.

HEAVYFIND Algorithm

The goal of the `HEAVYFIND` algorithm is to identify a set of frequent heavy vertices. Heavy vertices are defined as those with a degree at least half the maximum degree in the graph.

Algorithm 3 HEAVYFIND(H)

```
1: K ← H
2: i ← ⌈log |H|⌉
3: success ← FALSE
4: while success = FALSE do
5:    i ← i − 1
6:    if |{u | d_K(u) ≥ 2^i − 1}| ≥ |H|/⌈log |H|⌉ then
7:       success ← TRUE
8:    else
9:       K ← {u | d_K(u) < 2^i − 1}
10:   end if
11: end while
12: return K
```

Comments:

- We can assume that the graph is connected and can elect a leader.
- Each vertex then could ask the leader about $|H|$ and set $i = \lceil \log |H| \rceil$.
- We will also keep a flag whether the vertex is still in K or not.
- The algorithm proceeds in iterations, one iteration is:
  - Round 1: Vertices send their status (whether they are still in K) to neighbors and compute their degree.
  - Round 2: Vertices notify the leader about their degrees, and the leader determines whether the condition is met.
  - Round 3: Vertex receives the message from the leader and checks whether it should finish algorithm execution or not. If we are not finishing and our degree is at least $2^i - 1$ - we set the flag to 0.

SCOREFIND Algorithm

Let helper functions defined as $\text{prof}_K(u) = 1 + d_K(u)$
$\text{kill}_K(\{u, w\}) = 1 + \max(d_K(u), d_K(w))$
$\text{double}_K(\{u, w\}) = |N_K(u) \cap N_K(w)|$

$$\text{cost}_K(\{u, w\}) = \begin{cases} \text{kill}_K(\{u, w\}) & \text{if } \{u, w\} \in E(K) \\ \text{double}_K(\{u, w\}) & \text{if } \{u, w\} \notin E(K) \end{cases}$$

Then for every set $T \subseteq K$, define,

$$\text{Score}_K(T) = \sum_{u \in T} \text{prof}_K(u) - \sum_{\{u,w\} \subseteq T} \text{cost}_K(\{u, w\}).$$

If t = $|T|$, we also define:

$$\text{Rating}_K(R) = t \cdot \frac{1}{|R|} \sum_{u \in R} \text{prof}_K(u) - \binom{t}{2} \cdot \frac{1}{\binom{|R|}{2}} \sum_{\{u,w\} \subseteq R} \text{cost}_K(\{u, w\}),$$

The `SCOREFIND` algorithm identifies promising subsets of vertices based on scores and predefined criteria:

Algorithm 4 SCOREFIND(K)

$\ell \leftarrow \max\{\ell' \mid 2^{\ell'} - 1 \in [1, (|H|/\lfloor \log |H| \rfloor)]\}$
$n \leftarrow 2^\ell - 1$
$M \leftarrow$ an arbitrary set of m heavy vertices in K
for $u \in M$ **do**
    compute $\text{prof}_\kappa(u)$
end for
for each $\{u, w\} \subseteq M$ **do**
    compute $\text{cost}_\kappa(\{u, w\})$
end for
$\Delta \leftarrow$ maximum degree of a vertex in K
$s \leftarrow \max\{s' \mid 2^{s'} - 1 \in [1, \lceil m/(16\Delta) \rceil]\}$
$t \leftarrow 2^s - 1$
$U_\ell \leftarrow M$
for $j \leftarrow \ell$ **down to** $s + 1$ **do**
    Construct a block design with set of elements U and parameters:
    $v = 2^j - 1,\ r = k = 2^{j-1} - 1,\ \lambda = 2^{j-2} - 1$
    for each block R **do**
        Compute $\text{Rating}_K(R)$
    end for
    $U_{j-1} \leftarrow$ the block for which $\text{Rating}_\kappa(\cdot)$ is largest
end for
$T \leftarrow U_{j-1}$

Comments:

- Communication with the leader of H is required.
- Vertices notify the leader about every edge and statuses of them two.
- The leader processes the algorithm step-by-step, computing costs and profits, and notifies each vertex if they belong to T.

Overall Complexity

The algorithm's complexity is as follows:

- Number of iterations: $O(\log^2(n))$.
- Message complexity: $O(m^2 \cdot \log^2(n))$ (dominated by edge information sharing in each iteration).
- Time complexity: $O(n^3)$ (dominated by block rating calculations on the first iteration in leader).