Maximum Clique Approximation

Aaron Frye, Ben Lenox

November 2024

1 Overview

Approximating the maximum clique size in a graph is a difficult problem. The best solution in the literature is the Raymsey algorithm, as presented by Boppana and Halldòrsson. The algorithm works by randomly choosing a pivot node, then recursively finding the largest clique and independent set in the neighbors and non-neighbors of that node. The algorithm offers very little guarantee of performance, except in the case that the graph does not have any large independent sets.

2 Pseudocode

Algorithm 1 Ramsey algorithm

Inputs: A graph G and the subgraph sG that we search in
Outputs: An approximation of the size of the maximum independent set and the size of the smallest clique in G.

Ramsey(G, sG)
if $G == \varnothing$ **then**
  return $(\varnothing, \varnothing)$
end if
Choose $v \in sG$
$N(v) \leftarrow neighbors(G, v, sG)$
$\overline{N}(v) \leftarrow non\_neighbors(G, v, sG)$
$(I_1, C_1) \leftarrow$ **Ramsey**$(G, N(v))$
$(I_2, C_2) \leftarrow$ **Ramsey**$(G, \overline{N}(v))$
return $(\mathbf{max}(I_1,\ I_2\ \cup\ \{v\}),\ \mathbf{max}(C_1\ \cup\ \{v\},\ C_2))$

Here **max** returns the set with the higher cardinality. The notations $N(v)$ and $\overline{N}(v)$ for a vertex v refer to the set of nodes adjacent to v (neighbors) and the set of nodes not adjacent to v (non-neighbors) respectively.

3 Data Structures

The only data structures employed by the Ramsay algorithm are whatever data structures are used to represent $N(v)$ and $\overline{N}(v)$, in our implementation we use python sets. The paramater sG is one of these sets, representing the subgraph that we are currently searching.

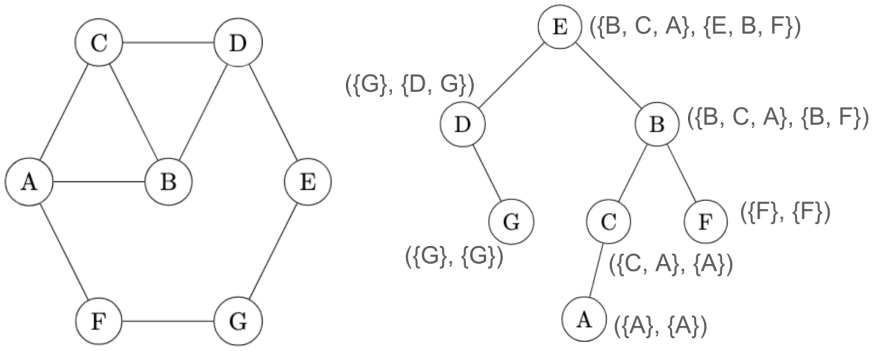
4 Intuition of Correctness

The following analysis is all due primarily to Boppana and Halldòrsson.

The Ramsey algorithm works by a simple approach of recursively building an optimal pair of sets based on a given pivot node.

First $v \in G$ is chosen, then we recursively run the algorithm on $N(v)$ and $\overline{N}(v)$. From our recursive calls we get back I_1 and I_2, the maximum independent sets found from looking in $N(v)$ and $\overline{N}(v)$ respectively. Since the nodes in I_2 are guaranteed not to be neighbors of v, we can add v to I_2 to produce a larger independent set. We finally compare I_1 with $I_2 \cup \{v\}$ and return the larger of the two. A similar process happens alongside this to compute the maximum clique,

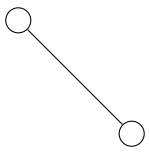
but instead of adding v when returning from $\overline{N}(v)$ we add it when returning from $N(v)$.



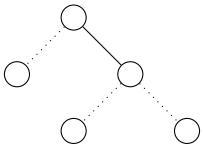
An execution of the Ramsay algorithm can be represented by a binary tree, as demonstrated in the above graphic. In this binary tree all left descendants of a given node are neighbors of that node, and all right descendants are non-neighbors. Note that if a clique with s nodes is found this will correlate to a path with $s-1$ left edges in the tree, and the same follows for independent sets with right edges, so the largest independent set or clique is determined by the path(s) with the maximum number of left edges or right edges.

Consider the function $r(s,t)$ which returns the smallest integer n such that any binary tree with n nodes is guaranteed to contain a path with at least $s-1$ left edges or $t-1$ right edges. Observe that n will be one larger than the number of nodes in the largest tree with no paths having $s-1$ left edges or $t-1$ right edges.

As an example, below is the largest graph with no path having 1 left edges or 2 right edges, used to find $r(2,3)$.



Below are all of the ways to add one node to this graph. Adding any of these nodes will result in a path with at least 1 left edge or 2 right edges.



The number of leaves in this tree is less than or equal to the number of unique paths in this tree which can be computed by the expression $\binom{s-1+t-1}{t-1} = \binom{s+t-2}{t-1}$. From this we conclude that $r(s,t) \leq \binom{s+t-2}{t-1}$.

The paper we read posits the following: the Ramsay algorithm produces a clique C and an independent set I for which $r(|C|, |I|) \geq n$, they do not specify what n refers to, but we will just say that it is some quantity associated with the graph.

We define the function $t_s(n) = min\{t | r(s,t) \geq n\} \leq min\{t | \binom{s+t-2}{t-1} \geq n\}$, which gives us for a specific s the minimum value of t that still guarantees $r(s,t) \geq n$. If our graph has no cliques of size $k+1$, and we know that the maximum value that our algorithm finds for $|C|$ is k. From our previous theorem, we know that our algorithm must have found $|C|$ and $|I|$ s.t. $r(|C|, |I|) \geq n$, and since $t_k(n)$ is the smallest clique for which this is the case, we know $|I| \geq t_k(n)$.

We then can approximate $t_k(n)$ and observe that the quantity of $|C| \cdot t_k(n)$ is maximized when they are equal, at which point each is larger than $\frac{1}{2}log(n)$, so $|I| \cdot |C| \geq \frac{1}{4}log(n)^2$.

5 Time Complexity

The Ramsey algorithm offers a divide and conquer approach to solving this problem, so we will analyze it's time complexity with a recurrence equation.

We first find the merge cost for a recursive call, which is determined by the approach we take for splitting up the graph into $N(v)$ and $\overline{N}(v)$. In our implementation $N(v)$ is derived by copying sG and then removing all nodes in sG which do not occur in the adjacency list of v. A similar process occurs for $\overline{N}(v)$. To do this we loop through u in sG, which takes n iterations, and in each iteration we check the membership of u in $adj[v]$, which takes roughly $O(1)$ time. Then if $u \notin adj[v]$ we remove it from $N(v)$, and if $u \in adj[v]$ we remove it from $\overline{N}(v)$, both of which take $O(1)$ time on average. The result is that our merge time is $O(n)$.

Since there can never be more than $n-1$ neighbors or non neighbors of a given node, we can derive an upper bound for the time complexity of our algorithm using the recurrence equation $T(n) = 2T(n-1) + n$. We posit that this recurrence is $O(2^n)$, and we show this by substitution. Assume that $T(n)$ is $O(2^n)$ for $n-1$, we will show the same for n:

$$T(n) = 2T(n-1) + n \tag{1}$$
$$\leq 2d2^{n-1} + n \tag{2}$$
$$= d2^n + n \tag{3}$$

With $d = 1/2$ we get:

$$2^{n-1} + n \leq 2^n \tag{4}$$

for $n \geq 3$.

6 Space Complexity

Our space complexity analysis is the same as our time complexity analysis, since we spend the same order of time allocating our data structures as we do merging.

7 Code

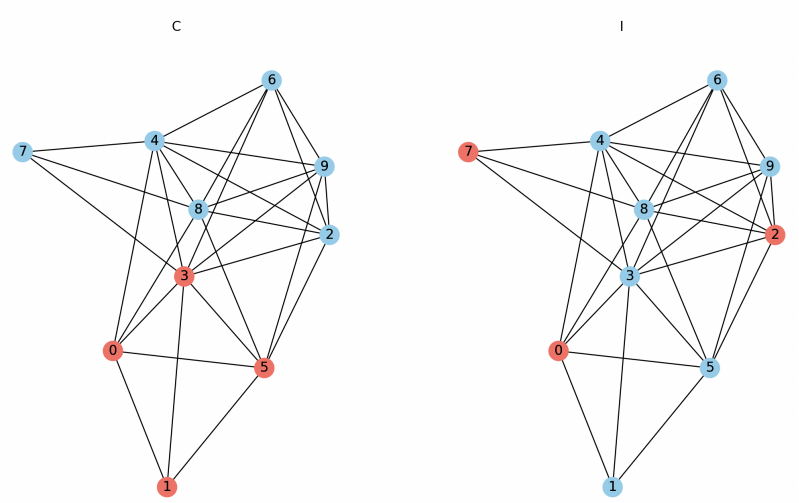
p4/p4.py

```python
import random
import networkx as nx
import matplotlib.pyplot as plt
import scipy

def Ramsey(G, sG):
    # O(1)
    if len(sG) == 0:
        return (nx.Graph(), nx.Graph())

    # O(1)
    v = sG.pop()
    adj = G.adj[v]

    # Neighbors
    N = sG.copy()
    # Non-Neighbors
    NC = sG.copy()

    # O(n)
    for u in sG:
        if u in adj:
            NC.remove(u)
        else:
            N.remove(u)

    # Recursive calls
    C1, I1 = Ramsey(G, N)
    C2, I2 = Ramsey(G, NC)

    # All O(1)
    if len(C1.nodes) + 1 > len(C2.nodes):
        C = C1
        C.add_node(v)
    else:
        C = C2

    if I2.number_of_nodes() + 1 > I1.number_of_nodes():
        I = I2
        I.add_node(v)
    else:
        I = I1

    return C, I



```

8 Results

8.1 Visualization

The graphs below indicate in red the Clique and independent set found by our algorithm for an input graph with 10 nodes and random edges.



8.2 Tests

Below are some sample tests of our algorithm on an input size of 1000 nodes. We expect to see that $|I| \cdot |C| \geq \frac{1}{4} log_2(1000)^2 \approx 25$ holds for our programs output, and in each case it does.

```
For graph with 1000 nodes and 0.3 chance of nodes being connected,

Without clique removal
Clique found: 8 nodes
Independent set found: 16 nodes

With clique removal
Clique found: 8 nodes
```

```
For graph with 1000 nodes and 0.5 chance of nodes being connected,

Without clique removal
Clique found: 10 nodes
Independent set found: 11 nodes

With clique removal
Clique found: 11 nodes
```

```
For graph with 1000 nodes and 0.8 chance of nodes being connected,

Without clique removal
Clique found: 24 nodes
Independent set found: 6 nodes

With clique removal
Clique found: 27 nodes
```

References

[1] Ravi Boppana and Magnús Halldórsson. "Approximating Maximum Independent Sets by Excluding Subgraphs". In: vol. 32. Jan. 2006, pp. 13–25. ISBN: 978-3-540-52846-3. DOI: 10.1007/3-540-52846-6_74.