CLIQUE NUMBER AND CHROMATIC NUMBER FOR PERMUTATION GRAPHS

ALEXANDER LEAF

1. Introduction

Let $[n_1, \cdots, n_k]$ be a permutation of $1, \cdots, k$. We can represent this permutation as a chord diagram on the circle with the following ordering:

$n_1, \cdots, n_k, k, \cdots, 1$

for which nodes with equal values are vertices for the chords. Let $a = n_i$ and $b = n_j$. Thus, when drawing the chord diagram to have the minimum number of crossings, we find that the chords connecting a to n_i and b to n_j will intersect only when $a > b$ and $i < j$ or when $a < b$ and $i > j$.

For any chord diagram, let the circle graph (or crossing graph) X_σ be the graph in which each chord is represented by a vertex and there is an edge between vertices if the two chords they represent cross.

Let the clique number ω be the size of the largest set of chords that intersects all other chords in the set and let a clique be such a set of chords. In the crossing graph, a clique corresponds to a complete graph and the clique number corresponds to the number of vertices in the largest complete subgraph. In other words, the mutually intersecting chord number is the size of the largest monotonically decreasing subsequence of the permutation.

Let a valid coloring of the chord diagram be any coloring of the chords such that no chords of the same color intersect each other. Let the chromatic number γ be the minimum number of colors that can be used in a valid coloring. In the chord diagram, this correspond to the chromatic number while coloring the vertices.

2. The Relationship Between Clique Number and Chromatic Number

Theorem 1. *Given permutation $\sigma = [n_1, \cdots, n_k]$ of $1, \cdots, k$ and its corresponding chord diagram, the clique number ω is equal to the chromatic number γ.*

Proof. It is clear that $\gamma \geq \omega$ because if $\omega > \gamma$, by the pigeonhole principle, at least two intersecting chords would have the same color, which would not be a valid coloring. To show that $\omega \geq \gamma$, we introduce the following greedy algorithm.

Algorithm 1. *The following algorithm finds cl for a given permutation. First, find the smallest unvisited value left in the permutation (at first 1) and mark that value as visited. Next, find the next smallest unvisited value n_j such that $n_j > n_i$ and $j > i$, let $i = j$ and mark n_j as visited. Repeat this process until there are no such values left. Next, on a new line, repeat this method of finding the smallest unvisited value and the subsequent values from the still unvisited values. This process ends when all values are visited.*

Date: July 19, 2010.

```
j = 0
M = [][] // the empty matrix
visited = [False, ...  , False]
while visited ≠ [True, ...  True]
    m := min k :  visited[k] = False
    i = 0
    while m ≠ ∞
       M[j, i] = m
       visited[m] = True
       m = min [ k > m : σ^{-1}(k) > σ^{-1}(m) AND visited[k] = false ]
       i := i + 1
    j := j + 1
```

Then, let the number of lines created be m. To find a maximal set of mutually intersecting chords from this list representation: Take the last number from line m. Then, for each preceding line i, take the largest number in the list smaller than the number selected from line i + 1.

```
m = max k :  M[k, 0] > 0
SET = [] // the empty array length m
SET[m] = M[m, max [k :  M[m, k] > 0]]
while m > 0:
    m := m - 1
    n := 0
    while M[m, n] < SET[m + 1]:
       n := n + 1
    SET[m] = M[m, n - 1]
```

Proof. For convenience, let the value selected from list i be l_i. By construction, the values in each list are monotonically increasing in the permutation and thus none of those chords intersect. Thus, we know that ω must be less than or equal to the number of lines. This means that we only have to show that the recovery algorithm always yields a set of chords and that those chords are all mutually intersecting.

First, assume that the recovery algorithm does not yield a set of chords. The only way this is possible is if for line i, l_{i+1} is less than all values in line i. However, this is impossible because by construction, the first value in line i is less than all values in lines j, $j > i$. Thus, the recovery algorithm will yield a set of chords.

Now, we just have to show that those chords are form a clique. By construction, $l_i, \cdots, l_\omega$ are monotonically increasing. Thus, it is sufficient to show that l_i comes after l_{i+1} in the permutation. If l_i is at the end of list i, then we know that it came after l_{i+1} in the permutation because the value from list i + 1 is greater, so l_i could not end list i if l_{i+1} came after it. If l_i is not at the end of list i, then by construction, the value after it in list i is greater than $l_i + 1$. Thus, l_i must come after $l_i + 1$ or else the value directly after l_i in list i would be less than $l_i + 1$, which could not be true. Thus, we have proven that l_i comes after $l_i + 1$ in the permutation and thus the chords formed from $l_1, \cdots, l_\omega$ form a clique. $\square$

End of proof of Theorem 1: If we use Algorithm 1, one valid coloring would be to color elements in the same line the same color. Thus, by coloring each line differently, no intersecting chords would have the same color. This would mean that $\omega \geq \gamma$. We already know that $\gamma \geq \omega$, so this means that $\omega = \gamma$, which proves the theorem. $\square$

3. Example

Take the following permutation [6, 5, 4, 7, 1, 2, 8, 3, 9] of $1, \cdots, 9$. Using our algorithm, we will start at 1, then take 2, 3 and because 4, 5, 6, 7 and 8 do not follow 3, we will take 9. Now, starting a new line, the lowest number left is 4, and because 5 and 6 do not come after it, we can take 7 and then 8. Nothing higher than 8 is left, so we move to a new line. Next, we take 5, and nothing higher comes after it, so we start a new line and take 6. Now, we have checked everything in the permutation. We should have the following:

1, 2, 3, 9
4, 7, 8
5
6

Thus, the minimum coloring is 4. Using the recovery algorithm, we first take the 6, then the 5. The 7 and 8 are greater than the 5, so we take the 4. Then, the 9 is greater than the 4, so we take the 3. Thus, we have 3, 4, 5 and 6 as our set.

4. Finding All Cliques

It is clear that any maximally intersecting chords must contain one element from each line or else, by the pigeonhole principle, two chords would not intersect, or we would have less than the maximum number of chords. This leads to the following recursive algorithm to find all maximal sets of mutually intersecting chords:

Algorithm 2. *We begin with the line format from Algorithm 1. We begin on the last line (as with the recovery algorithm above). However, this time, we take all elements from the last line. We use the following recursion:*

Given an element e on line i $i > 1$, recurse on all elements in line $i - 1$ such that the elements are less than e and come after e in the permutation.

```
m = max [ k :  M[k, 0] > 0 ]
ST = empty // empty symbol table holding arrays
SET = [] // empty array length m
recurse(m, ∞,  SET)

recurse(n, last, SET):
list = all nonzero values on line n
for all [ v in list :   (σ^{-1}(v) > σ^{-1}(last) AND last > v) OR last = ∞  ]
    SET[n - 1] = v
    if n > 1:  recurse(n - 1, v, SET)
    else:  add SET to ST
```

All sets covered in the recursion are mutually intersecting chords of maximal size.

Proof. We know from the proof of the first recoverability algorithm that we will get at least one valid element on line $i-1$ given an element on line i. The proof that all sets created will be mutually intersecting chords follows from the proof of the first recoverability algorithm. Thus, we just have to prove that we will get all mutually intersecting chords of maximal size with this algorithm. Assume that there is a set of maximally intersecting chords that this algorithm doesn?t give us. Because all elements on the final line are considered, there must be a first line i going backwards such that the element selected from the line above is not considered. This would mean that either: 1) The element on line $i-1$ is larger than that from line i (e). This implies that we can find a minimum element on line $i-1$ greater than e. We know that below this on line $i-1$ there is an element less than e, which means that all of these elements come after e in the permutation, so the chord from line i and line $i-1$ would not intersect, which cannot be true. 2) The element on line $i-1$ is smaller than e but comes before it in the permutation. But this would also mean that the chord from line i and line $i-1$ would not intersect, which cannot be true.

Thus, this set would not be a set of maximally intersecting chords, and thus the algorithm finds all maximally intersecting chords. □

5. Significance

These permutation representing chord diagrams are a special case of circle graphs. For a general circle graph, chromatic number is not always equal to clique number. For example, the 5-cycle can be given as a circle graph, for which a corresponding chord diagram could have vertex ordering [1, 2, 3, 4, 2, 5, 4, 1, 5, 3]. In this case, chromatic number would be 3, while clique number would be 2. For a general circle graph, finding the chromatic number and maximum independent set size is hard, while finding clique number and a maximum clique and maximum independent set has worst-case $O(N^3)$ time. (For permutation graphs, this process has worst-case $O(N^2)$ time.) For this general algorithm, see Gavril 1973 (Algorithms for a Maximum Clique and a Maximum Independent Set of a Circle Graph). One possible use for this algorithm is finding trivializing number for a knot pseudodiagram (see Henrich et al. 2009 (Classical and Virtual Pseudodiagram Theory and New Bounds on Unknotting Numbers and Genus)).