CSCI-761 Combinatorial Computing
Assignment 5
Due Tuesday, March 31, 2026

Name: Tiffany Lee

1. Pruning Backtracking

CNF-SAT is a problem of finding satisfying assignments to a Boolean formula α in conjunctive normal form, i.e. where α is given by a set of clauses, each of which is a disjunction of literals, each of which is a variable or a negated variable. Suppose you are solving instances of CNF-SAT by backtracking. Describe informally a bounding function (in the spirit of bounding functions used in the maximum clique search algorithms considered in class) which could be used to prune the search space for satisfying assignments to instances of CNF-SAT.

Almost everybody is using the so called DIMACS-CNF format for en coding propositional formulas in the CNF form, see for example:
people.sc.fsu.edu/~jburkardt/data/cnf/cnf.html, or
users.aalto.fi/~tjunttil/2021-DP-AUT/notes-sat/solving.html.

To follow the struggles of many researchers trying to do it better and better already for decades, explore http://www.satcompetition.org, which is a website of worldwide competitions for designing state-of-art SAT-solvers.

Suppose you were backtracking through a CNF-SAT instance. At any partial assignment, I would break up the clauses into three categories:

Already satisfied, undecided, or already unsatisfiable.

If a clause is already unsatisfiable then every literal in that clause has already been assigned and they are all false. Then you could use an optimistic upper bound. The most number of clauses this branch can possibly satisfy is the number of already satisfied clauses + number of undecided clauses. If there is at least one clause that is already unsatisfiable, then you know this upper bound is strictly less than the total amount of clauses. So you will never be able to reach a satisfying assignment from this point. If we reached this scenario we would prune the branch.

A more strict way of doing the same thing is to use unit clauses. Continuously apply the forced assignments from unit clauses under the current partial assignment. If that propagation causes an empty clause to be created, then we know the upper bound has dropped to 0 for total satisfiability. So then we can cut that branch. Informally the bounding is asking "is it possible for all clauses to be satisfied from here?" If not then we don't need to explore that branch any further.

2. Ramsey/Folkman numbers

(a) Which of the following are true? Give yes/no answer and a brief justification for each.

- $C_5 \rightarrow (3, 3)^e$
  - No, because C_5 has no triangle at all so no edge coloring can produce a monochromatic K_3.
- $C_5 \rightarrow (3, 3)^v$
  - No, because C_5 contains no K_3, so a monochromatic triangle is not possible.
- $C_5 \rightarrow (2, 2)^v$
  - Yes, because this is asking whether every 2 color vertex coloring gives a monochromatic edge and since C_5 is an odd cycle, it is not bipartite.
- $C_5 \rightarrow (2, 2, 2)^v$
  - No, since C_5 is 3 colorable, meaning there is a proper 3 color vertex coloring with no monochromatic edge.
- $K_4 \rightarrow (3, 3)^e$
  - No, because there is a 2 edge coloring of K_4 with no monochromatic triangle, such as color a 4 cycle red and the two diagonals blue
- $K_5 \rightarrow (3, 3)^e$
  - No, because if you color the edges of a 5 cycle red and the remaining edges blue, neither color contains a triangle.
- $K_6 \rightarrow (3, 3)^e$
  - Yes, this is exactly the classical fact $R(3, 3) = 6$
- $K_5 \rightarrow (3, 3)^v$
  - Yes, because any 2 color vertex coloring of 5 vertices has at least 3 vertices that are the same color, therefore those three vertices must form a K_3 since the graph is complete.
- $K_5 \rightarrow (2, 2, 2)^v$
  - Yes, because K5 needs 5 colors to be properly vertex colored, forcing any coloring that uses only 3 colors to have a monochromatic edge.
- $K_5 \rightarrow (2, 2, 2, 2)^v$
  - Yes with similar reasoning as the statement above. K_5 cannot be made 4 colorable without having a monochromatic edge.
- $K_5 \rightarrow (2, 2, 2, 2, 2)^v$
  - No, if each vertex were a different color there would not be any monochromatic edge.

(b) Is it easier to prove a lower or upper bound for a Ramsey number? Is it easier to prove a lower or upper bound for a Folkman number? Explain your answers.

For Ramsey numbers, it is typically easier to prove lower bounds. This is because proving a Ramsey number is at least N only requires exhibiting a single coloring or witness graph demonstrating that a monochromatic clique is still avoidable. Proving the upper bound is harder since it requires showing that every coloring on that many vertices must contain the desired monochromatic clique. In the lecture slides, this is framed as the difference between finding a witness versus proving that no witness exists, with proving there exists no witness described as the hard exhaustive part. For Folkman numbers, the situation is reversed. Given that a Folkman number represents the smallest possible graph size exhibiting a certain arrowing property, establishing upper bounds is a more manageable task, since it only requires the presentation of such a graph. Lower bounds are more difficult to prove since you must show that all graphs of smaller order do not have the property. This usually involves considerably more case analysis or computation.

(c) Prove that k > R(s, t) implies Fe(s, t; k) = R(s, t). Hint: It is not hard, just combine the definitions.

Assume $k > R$ (s,t). By definition of Ramsey number $K_{R(s,t)} \rightarrow (s,t)^e$. $k > R(s, t)$ implies $K_{R(s, t)}$ does not contain K_k. So $K_{R(s, t)}$ satisfies the property in the definition of $F_e(s, t; k)$ which means $F_e(s, t; k) \leq R(s, t)$. Now assume $F_e(s, t; k) < R(s, t)$, let G be some graph on fewer than $R(s, t)$ vertices such that $G \rightarrow (s, t)^e$. Then $K_n \rightarrow (s, t)^e$ also holds for the complete graph on the same vertex set as G because every coloring of Kn restricts to a coloring of G. However, this would contradict the definition of $R(s, t)$ as the least order such that $K_n \rightarrow (s, t)^e$ for all $n \geq R(s, t)$. Therefore $F_e(s, t; k) \geq R(s, t)$. Now by the trivial inequality $F_e(s, t; k) \leq R(s, t)$ we have $F_e(s, t; k) = R(s, t)$.

(d) Prove that $K_3 + C_5 \rightarrow (3, 3)^e$.

Let the vertices of the K_3 piece be named a, b, c. Assume there is some red blue coloring of $K_3 + C_5$ with no monochromatic triangle. We will derive a contradiction. Since the triangle formed by a, b, c can't be monochromatic we may assume without loss of generality that ab and ac are red and bc is blue. Now consider any vertex x on the C_5. If ax is red then both bx and cx must be blue in order to avoid a red triangle. But then we would have a blue triangle bcx which would be impossible. Therefore all of the edges from vertex a to the C_5 must be blue. Now every edge in the C_5 must be red. Assume for contradiction that there is some edge xy in the cycle that is blue. But ax and ay are both blue, therefore axy is a blue triangle which is impossible. Consider the 5 vertices of the C_5 now. Note that either bx or cx is red otherwise bcx would form a blue triangle. Since there are 5 vertices on C_5 either b or c sends red edges to at least 3 vertices of the C_5. Without loss of generality assume that b does. Now take any 3 vertices from the C_5. Two of them must be consecutive on C_5, say x and y. xy is an edge of C_5 so it is red. Also bx and by are both red, so bxy is red triangle and we have obtained a contradiction. Every red blue coloring of $K_3 + C_5$ has a monochromatic triangle. So $K_3 + C_5 \rightarrow (3, 3)^e$

3. Progress on Folkman problems

In 2014, Christopher Wood wrote a comprehensive survey of the area, at https://www.cs.rit.edu/~spr/COURSES/CCOMP/cawfolk.pdf. In 2018, Aleksandar Bikov defended his PhD dissertation titled ”Com putation and Bounding of Folkman Numbers”, available at https:// arxiv.org/abs/1806.09601 (both are also linked from the item 14 on the course page). List four results which are in the latter but not in the former.

(a) $F_v(2, 2, 6; 7) = 17$

Bikov gives this as an exact value in the dissertation. In Wood’s 2014 survey, this exact result is not listed, and the earlier information only gives the weaker upper bound $F_v(2, 2, 6; 7) \leq 22$

(b) $F_v(3, 6; 7) = 18$

This appears in the dissertation as an exact computation. In the 2014 survey, it was still only recorded as $F_v(3, 6; 7) \leq 18$, so the exact value is a newer result there.

(c) $F_v(2, 2, 7; 8) = 20$

Bikov computes this exactly in the dissertation. The 2014 survey does not give the exact value and only had the earlier upper bound $F_v(2, 2, 7; 8) \leq 28$.

(d) $20 \leq F_v(3, 7; 8) \leq 21$

This tighter bound is included in the dissertation as a newer result. In Wood’s survey, the corresponding entry was still weaker, with only the upper bound $F_v(3, 7; 8) \leq 22$

4. Making Graphs

Make nauty canonical labelings of the following five graphs. In each case describe the steps how you made them.

(a) The Petersen Graph : IsP@OkWHG

```python
# 1.) Generated Petersen Graph using nauty's genspecialg fucntion
./genspecialg -g -P5,2 > petersen.g6

# 2.) Used labelg fucntion to apply the nauty canonical label to the graph
./labelg petersen.g6 > petersen_labeled.g6

# 3.) Used the cat function to print the canonically labelled graph
cat petersen_labeled.g6
```

(b) The Gr¨otzsch Graph : J?AKagjXfo?

```python
# 1.) Generated 11 vertex 20 edge graph candidates using nauty's geng function,
then used pickg to filter down to the Grötzsch Graph by requiring it to be
triangle free, have automorphism group size 10, have radius 2, and diameter 2
./geng 11 20:20 | ./pickg -T0 -a10 -z2 -Z2 > grotzsch.g6

# 2.) Used labelg function to apply the nauty canonical label to the graph
./labelg grotzsch.g6 > grotzsch_labeled.g6

# 3.) Used the cat function to print the canonically labelled graph
cat grotzsch_labeled.g6
```

(c) The Clebsch Graph : OsaBA`GP@`dIHWEcas_]O

```python
# 1.) Generated the Clebsch Graph candidates using nauty's geng function and filtered to the one with the right properties
./geng -t 16 -d5 -D5 40:40 | ./pickg -z2 -Z2 -a1920 > clebsch.g6

# 2.) Used labelg function to apply the nauty canonical label to the graph
./labelg clebsch.g6 > clebsch_labeled.g6

# 3.) Used the cat function to print the canonically labelled graph
cat clebsch_labeled.g6
```

(d) The Paley Graph P17 : P}qTKukXaUja[IBjanPeMI\K

```python
# 1.) Generated the Paley Graph P17 using nauty's genspecialg function with the -Y17 Paley Graph option, then used labelg to apply the nauty canonical label in graph6 format
./genspecialg -g -q -Y17 | ./labelg -q > paley17_labeled.g6

# 2.) Used the cat function to print the canonically labelled graph
cat paley17_labeled.g6
```

(e) The Schl¨afli Graph :
Z~~vnZjvUtw~nSmis{{k~a^||QBtQJNHLU[VQ^BxkFnDK\zEEvn@Tn^_Tn^w

```python
# 1.) Generated the Schläfli Graph using nauty's genspecialg named graph option, then used labelg to apply the nauty canonical label in graph6 format
./genspecialg -q -XSchlaefliGraph | ./labelg -g -q > schlafli_labeled.g6

# 2.) Used the cat function to print the canonically labelled graph
cat schlafli_labeled.g6
```

(f) The Nenov Graph for Fv(24; K4) on 11 vertices : J`?EHu^Zvx_

```python
# 1.) Generated all 11 vertex K4-free graphs using nauty's geng function, then used pickg to keep only the graphs with chromatic number 5
./geng -g -q -k 11 | ./pickg -N5 > nenov_family.g6

# 2.) Used labelg function to apply the nauty canonical labels to the resulting graphs
./labelg -g -q nenov_family.g6 > nenov_family_labeled.g6

# 3.) Used the cat function to print the canonically labelled graphs, then selected one valid graph from the family
cat nenov_family_labeled.g6
```