3-Coloring in time $O(1.3446^n)$: a no-MIS algorithm

Richard Beigel

and

David Eppstein

Why try to solve graph coloring exactly?

- With fast computers we can do exponential-time computations of moderate and increasing size
- Algorithmic improvements are even more important than in polynomial-time arena
- Graph coloring is useful e.g. for register allocation and parallel scheduling
- Approximate coloring algorithms have poor approximation ratios
- Interesting gap between theory and practice

Previous 3-coloring methods

- Color vertices one at a time,
  ordered by fewest available choices:
  2^n [folklore?]

- For each maximal independent set
  test if remaining graph is bipartite:
  $3^{n/3} \approx 1.4422^n$ [Lawler 1976]

- Use maximal independent sets to increase
  vertex degree or split into subproblems:
  1.415^n [Schiermeyer 1994]

Our method

- Replace by a more general problem:
  symbol system satisfiability (3,2)-SSS

  Idea: more flexibility for local reductions to stay within the same problem class

- Solve (3,2)-SSS by finding unavoidable set of reducible local configurations

  Idea: similar strategy to proof of 4-color theorem

Result: 1.3803^n

- Improved reduction 3-coloring $\Rightarrow$ (3,2)-SSS

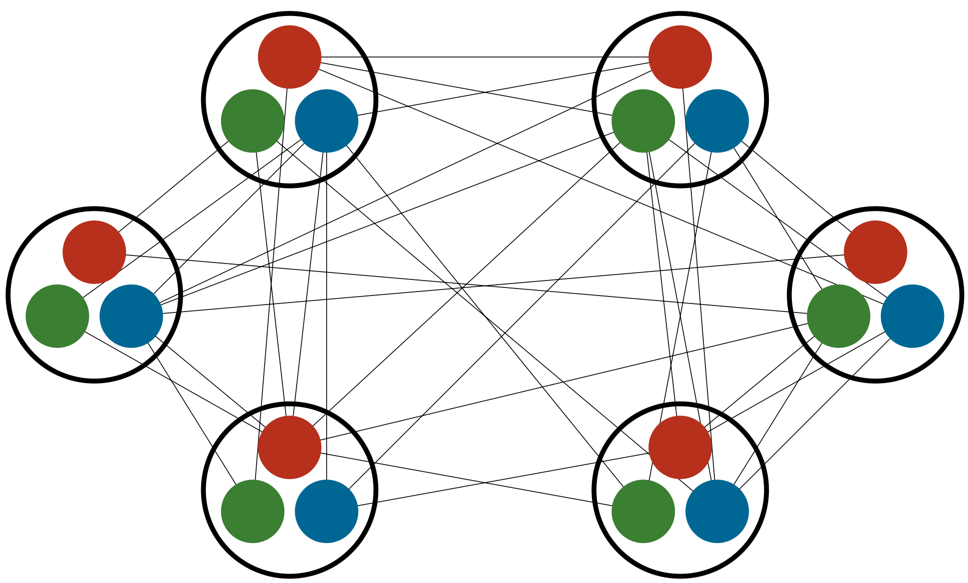
  Idea: choose colors for a few high-degree vertices then solve remaining (3,2)-SSS problem

Result: 1.3446^n

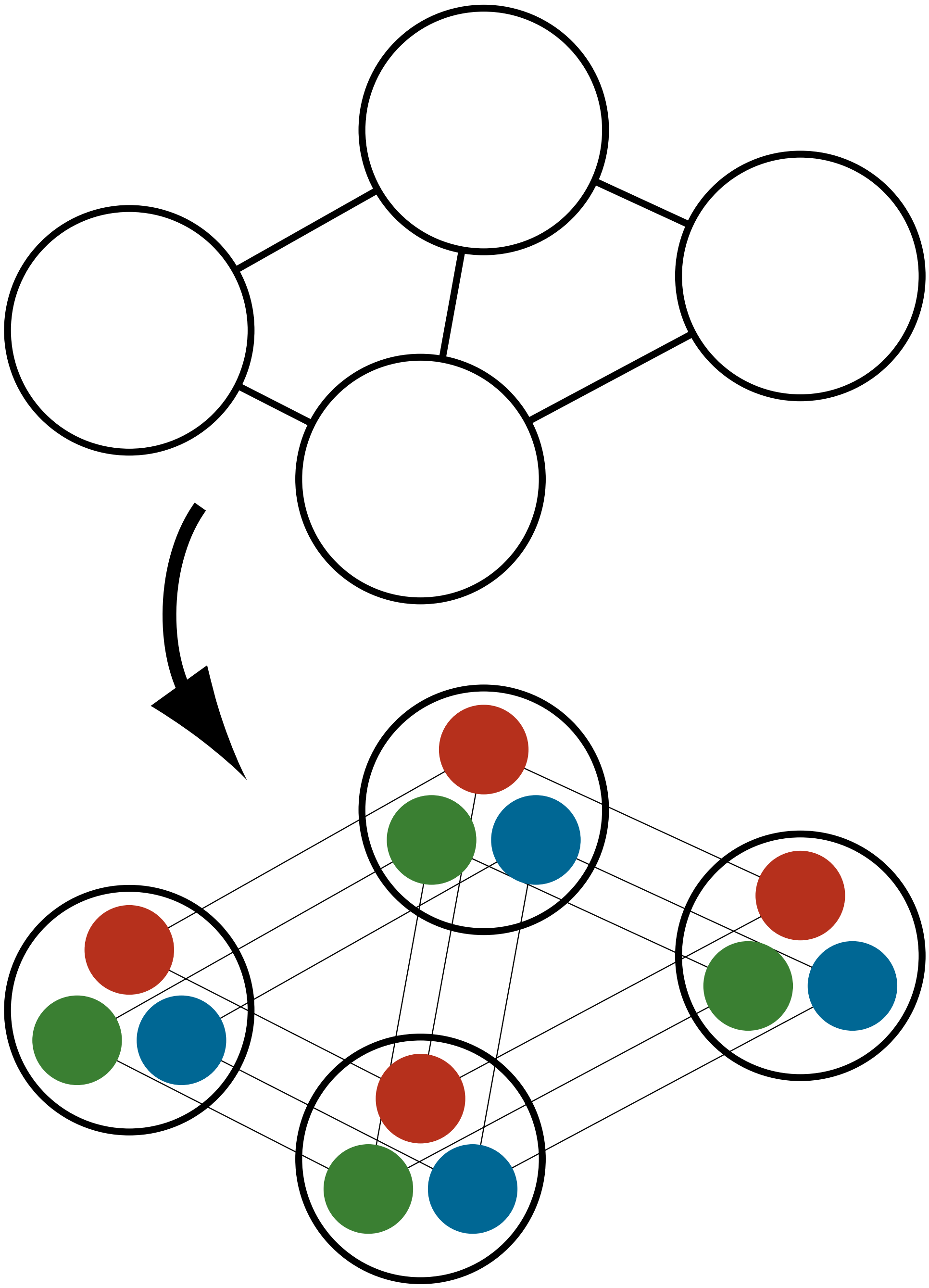
What is (3,2)-SSS?

(3,2)-SSS

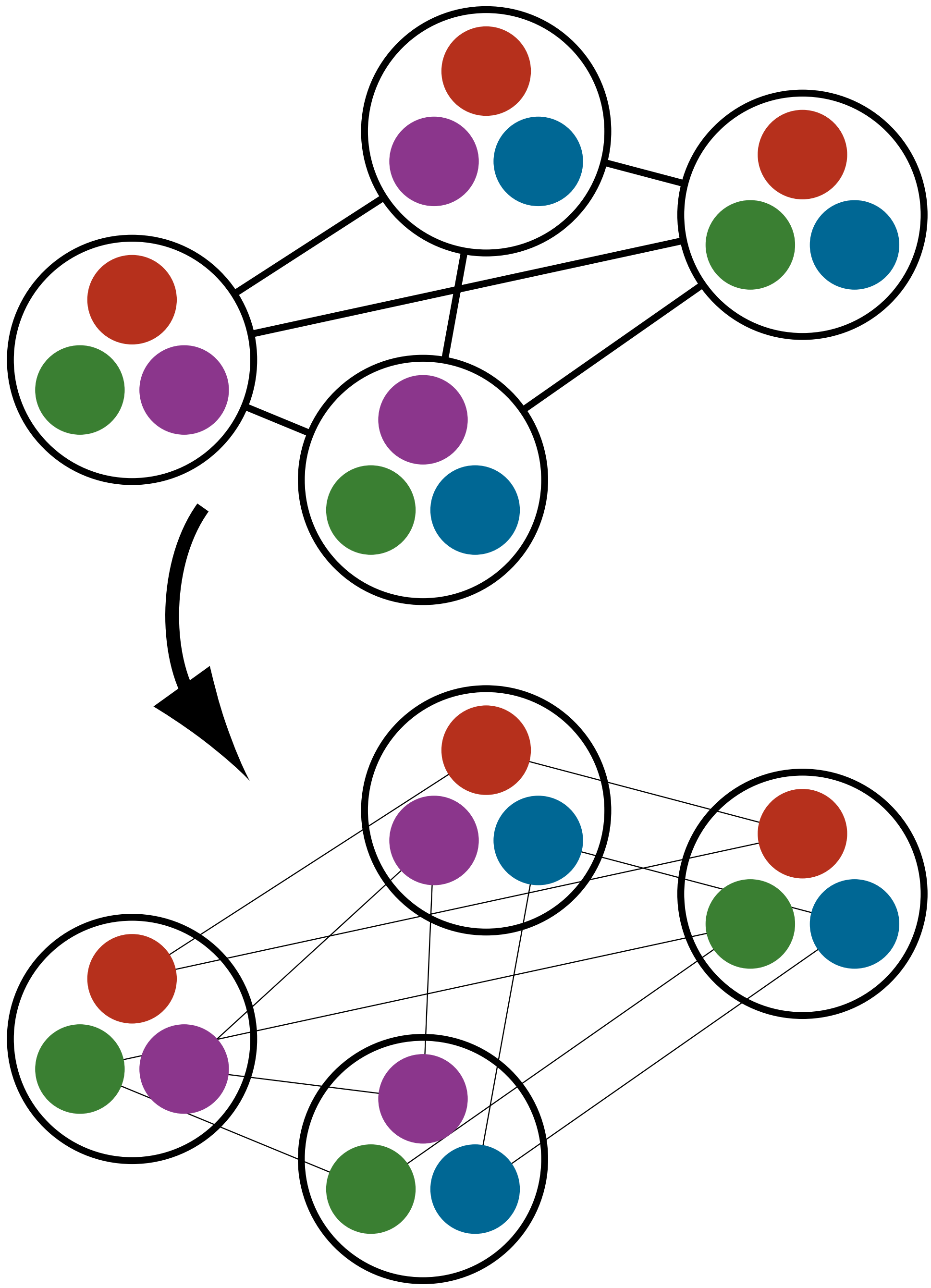
- Set of vertices (variables)
- Three colors (values) per vertex
- Edges (constraints) between incompatible pairs of colors



- Color all vertices without incompatibilities

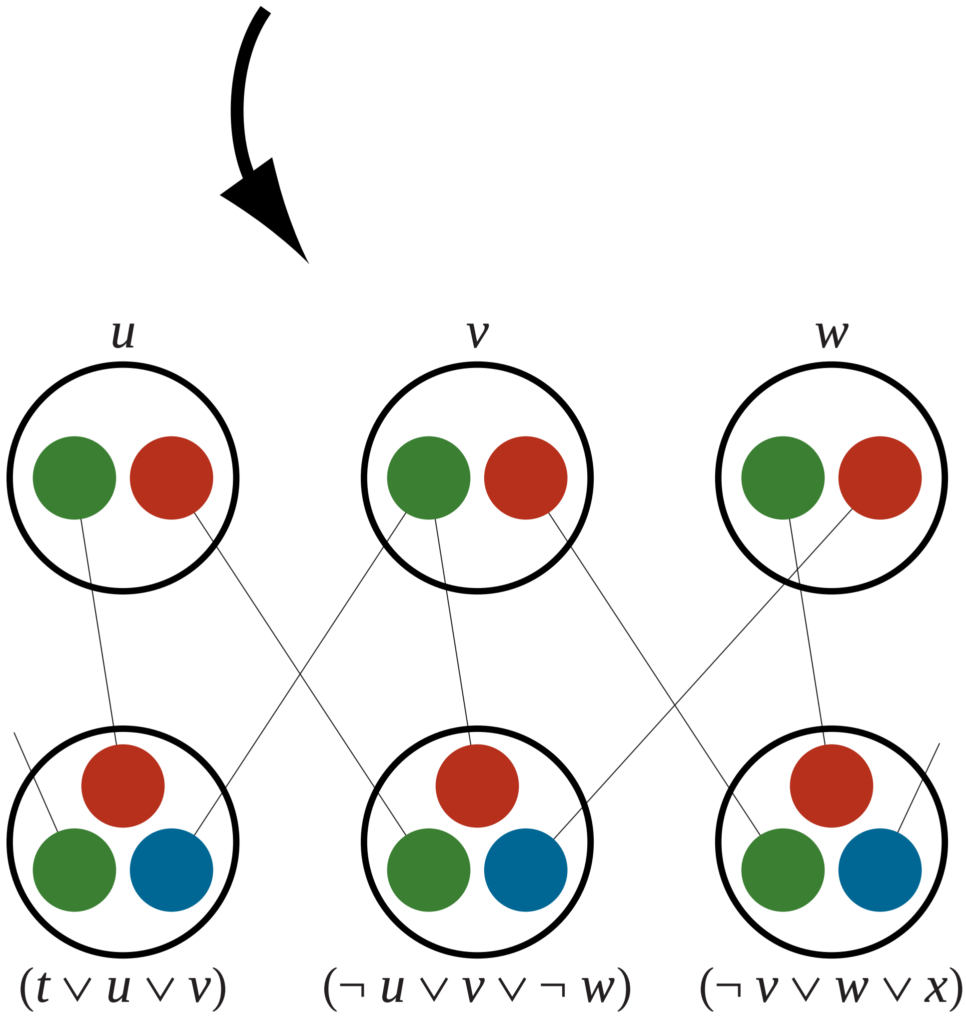


3-coloring $\Rightarrow$ (3,2)-SSS



3-list-coloring $\Rightarrow$ (3,2)-SSS

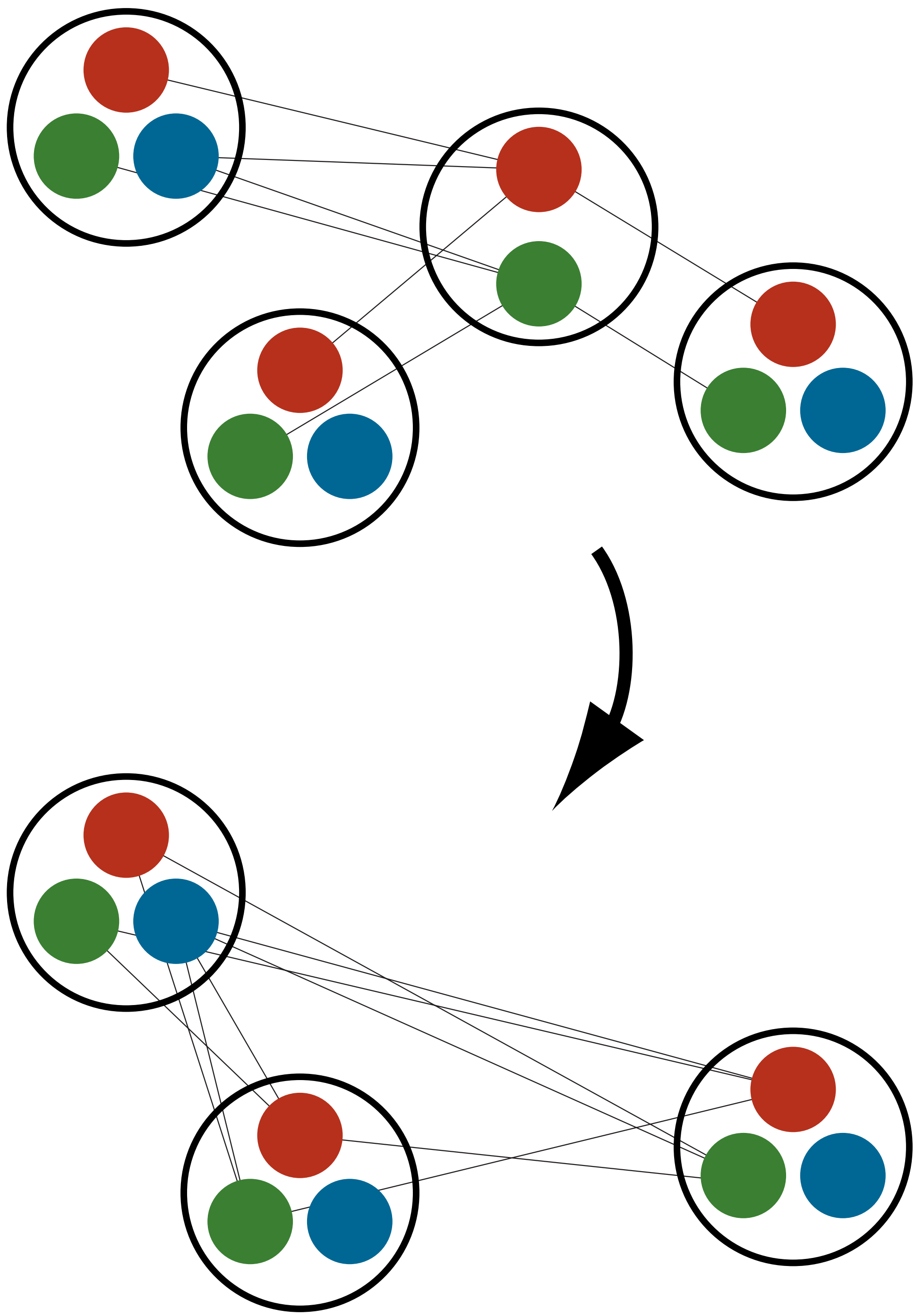
$(t \lor u \lor v)\ (\neg u \lor v \lor \neg w)\ (\neg v \lor w \lor x)$



3-SAT $\Rightarrow$ (3,2)-SSS

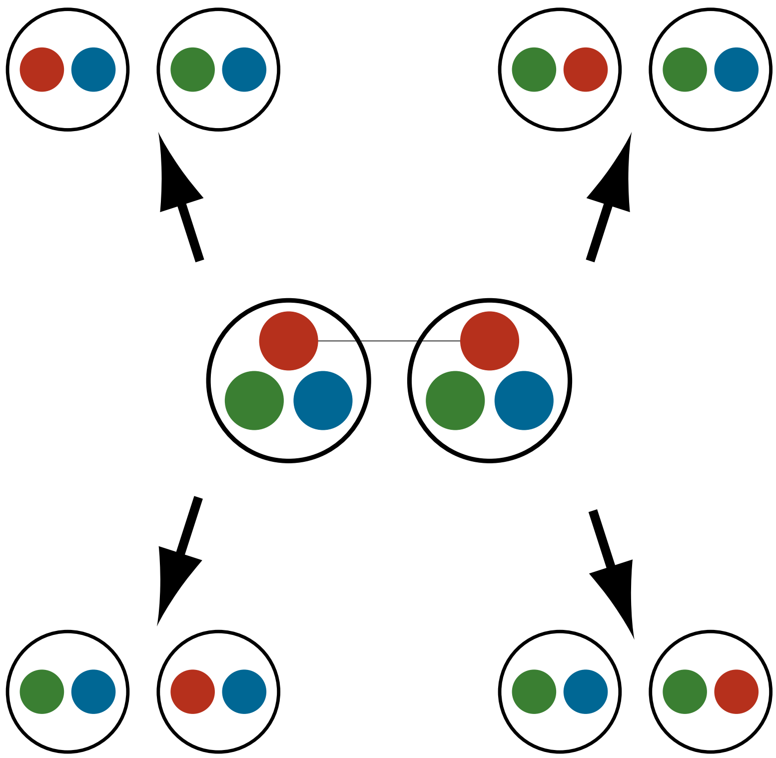
How do we solve (3,2)-SSS?

Vertices w/only two colors are free!



Simple $2^{n/2}$ algorithm

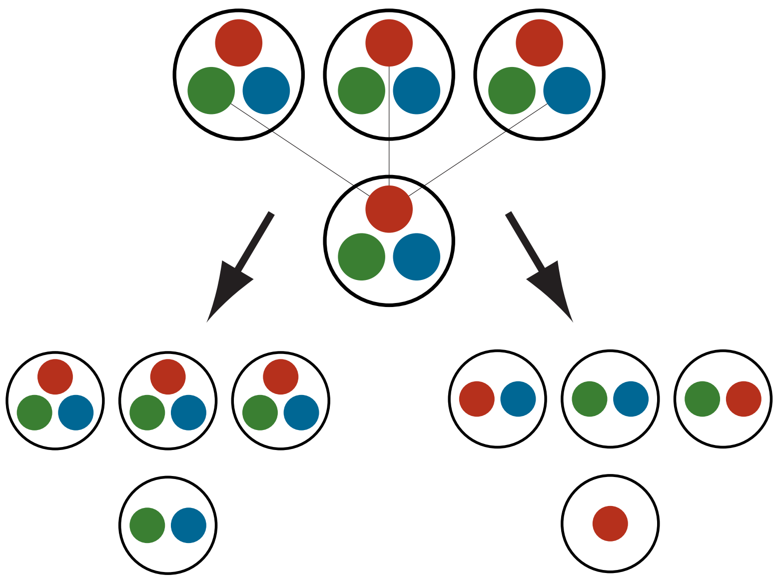
- Randomly restrict two adjacent vertices
- Four possible restrictions using exactly one of the two incompatible colors



- 50% chance of preserving consistent coloring
- Reduces problem size by two vertices

Deterministic 1.3803^n algorithm

- Messy case analysis
- Main case: some vertex has a color with at least three neighbors



- Restricting to remaining colors removes one vertex
- Using that color removes four vertices
- $T(n) = T(n-1) + T(n-4) = 1.3803^n$

Remaining Cases

- Colors with multiple neighbors
  in the same neighboring vertex

- Colors with only a single neighbor

- Long chains of degree-two colors

- Short cycles of colors

- If all other cases exhausted,
  only triangles of colors remain—
  solvable by Hall's Theorem!

Bushy Forests

or,

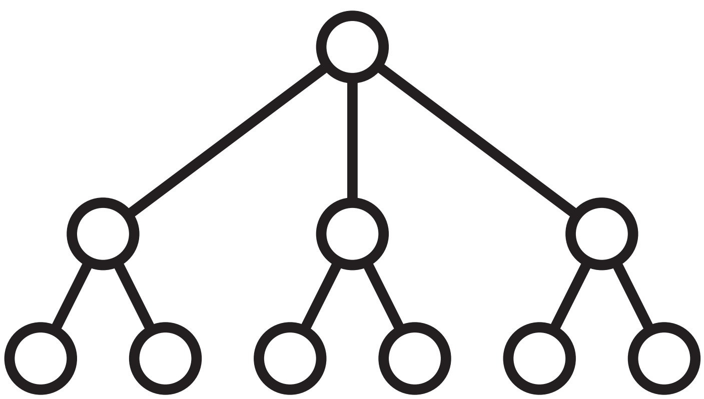
reducing 3-coloring to (3,2)-SSS

Idea:

- Find set S of high degree vertices
- Choose a color for each member of S
- Treat remaining vertices as (3,2)-SSS problem
- Each neighbor of S is restricted to two colors and eliminated
- If S small but $N(S)$ large, cost of coloring S more than made up by savings of eliminating $N(S)$

Basic Reduction Technique

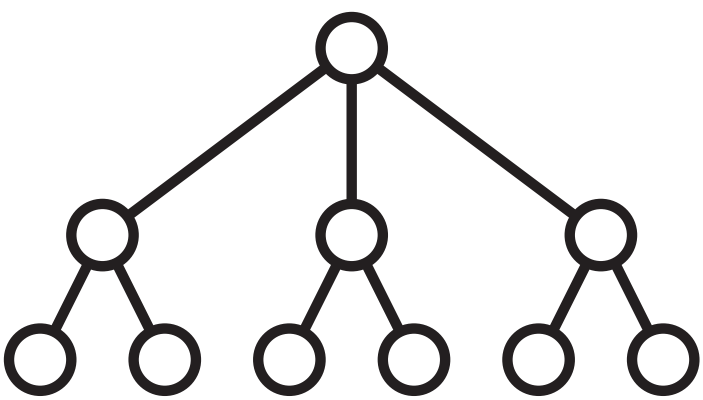
- Find maximal set of vertices
  with no shared neighbors
- Forest of shortest paths to set has height two
- Color each tree root and degree-≥ 3 child
- Worst case: three children, six grandchildren



- Choosing root color eliminates four vertices
- Remaining six grandchildren $\Rightarrow$ (3,2)-SSS
- Cost per vertex: $(3 \cdot 1.3803^6)^{1/10} \approx 1.3542$

Improved Reduction Technique for three children, two w/degree ≥ 2

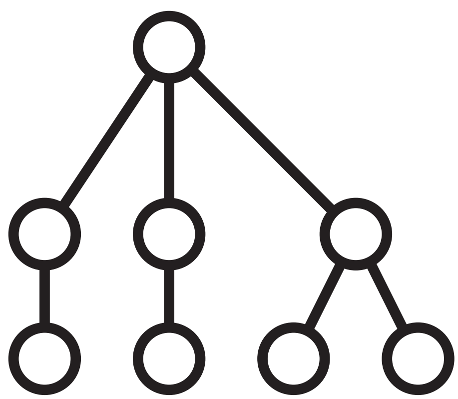
- Color two children in each of nine ways
- If children have different colors color of tree root is forced and third child is eliminated
- If same color, third child $\Rightarrow$ (3,2)-SSS
- Same worst case:



- Cost per vertex: $(6 \cdot 1.3803^2 + 3 \cdot 1.3803^3)^{1/10} \approx 1.3446$

Improved Reduction Technique for other trees

- Color root and bushy children as before
- Worst case: tree with four grandchildren



- Cost per vertex: $(3 \cdot 1.3803^4)^{1/8} \approx 1.3478$
- Eliminate these bad trees
  (local improvement, messy case analysis, complicated potential function)
- Worst remaining tree: three grandchildren
  cost $= (3 \cdot 1.3803^3)^{1/7} \approx 1.3432$

Conclusions

- New faster algorithm for 3-coloring
- Some improvement possible by more complicated case analyses
- Is c^n the right form of time bound?
- How can we find the right value for c?