Determining a Graph's Chromatic Number for Part Consolidation in Axiomatic Design

Jeffery A. Cavallaro

San Jose State University
Department of Mathematics

27 March 2020

Axiomatic Design

- Formalizes the design process without affecting creativity.
- Attempts to identify those traits common to successful designs.
- Starts with a set of *functional requirements* (FRs) that address customer needs.
- Designers construct *design parameters* (DPs) to satisfy the FRs.
- Provides a framework for comparing different designs.

The Axioms

The Independence Axiom

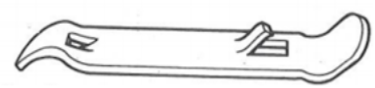
An optimal design always maintains the independence of the FRs. This means that the FRs and DPs are related in such a way that a specific DP can be adjusted to satisfy its corresponding FR without affecting other FRs.

The Information Axiom

The best design is a functionally uncoupled design that has the minimum information content.

Part Consolidation

- Minimize information content by consolidating multiple FRs and their DPs into a single part.
- The minimum number of parts needed to satisfy all FRs is a key metric for comparing designs.
- Opener example: 2 FRs (open bottles, open cans) and 1 part.



A Graph Theory Solution

- Let the FRs be vertices in a simple graph.
- If two FRs cannot be combined into the same part for some reason then add an edge between their vertices.
- The minimum number of parts problem becomes a chromatic coloring problem of the resulting graph.

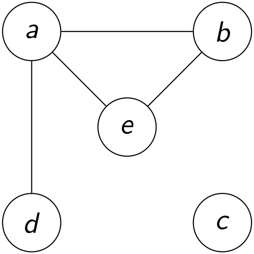
The Chromatic Coloring Problem

- Inherently intractable (steps/time required to solve increases exponentially with order, NP-hard).
- P-time algorithms to estimate: not exact but maybe a good start.
- Exact algorithms:
  - Christofides
  - Zykov
  - Jahanbekam/Cavallaro (proposed by this research)
- Solution Parameters:
  - Approximately 20 FRs (vertices).
  - Moderate edge density.
  - Runtime duration of under one minute.

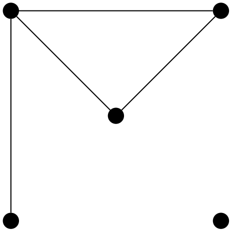
Simple Graphs

- A mathematical object $G = (V, E)$ that includes a set of vertices (nodes) $V(G)$ and and set of edges $E(G)$.
- Each edge is a 2-element subset of $V(G)$: $E(G) \subset \mathcal{P}_2(V(G))$.
- Edges are identified by juxtaposition: $\{a, b\} = ab = ba$.
- No multiple edges and no loops.
- Vertices can be labeled or unlabeled.

Simple Graph Example



LABELED



UNLABELED

$$V(G) = \{a, b, c, d, e\}$$
$$E(G) = \{ab, ad, ae, be\}$$

Adjacent Vertices

- Vertices joined by an edge are *adjacent* or *neighbors*.
- An edge *joins* and is *incident* to its vertices.
- An *isolated* vertex has no neighbors.

Graph Order and Size

- The *order* of a graph is the number of vertices: $n = |V(G)|$.
- The *size* of a graph is the number of edges: $m = |E(G)|$.
- $m \leq \binom{n}{2} = \frac{n(n-1)}{2}$

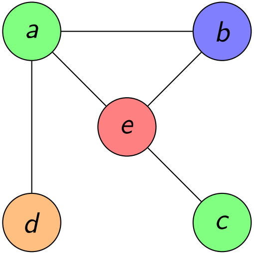
Order/Size Special Cases

- The *null* graph has no vertices ($n = m = 0$).
- An *empty* graph has no edges (m=0).
- A *complete* graph has every possible edge $\left(m = \dfrac{n(n-1)}{2}\right)$.

Graph Coloring

- A *coloring* of a graph is a function $c : V(G) \to C$ that assigns a color from C to each vertex.
- A *proper* coloring of a graph assigns different colors to adjacent vertices: $uv \in E(G) \implies c(u) \neq c(v)$.
- The coloring function need not be surjective.
- A proper coloring with $|C| = k$ is called a k-coloring.
- A k-colorable graph is also $(k+1)$-colorable.
- If $n \leq k$ then a graph is guaranteed to be k-colorable.
- A coloring for a minimum k is called a *chromatic* coloring.
- The minimum such k is called the *chromatic number* of a graph: $\chi(G)$.

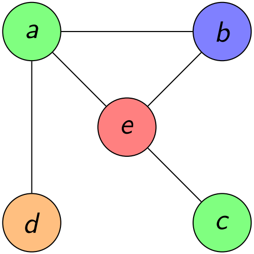
4-coloring Example



$$C = \{\text{green}, \text{blue}, \text{red}, \text{orange}\}$$

$$c(a) = \text{green}$$
$$c(b) = \text{blue}$$
$$c(c) = \text{green}$$
$$c(d) = \text{orange}$$
$$c(e) = \text{red}$$

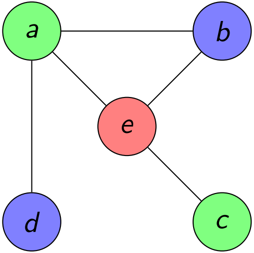
5-coloring Example



$$C = \{\text{green}, \text{blue}, \text{red}, \text{orange}, \text{brown}\}$$

$$\begin{aligned} c(a) &= \text{green} \\ c(b) &= \text{blue} \\ c(c) &= \text{green} \\ c(d) &= \text{orange} \\ c(e) &= \text{red} \end{aligned}$$

Chromatic Coloring Example

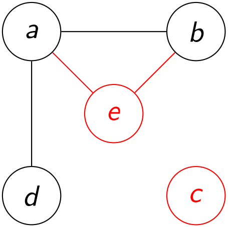


$$C = \{\text{green}, \text{blue}, \text{red}\}$$

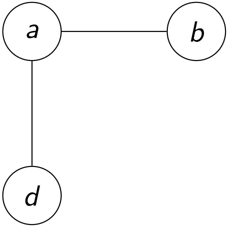
$$\begin{aligned}
c(a) &= \text{green} \\
c(b) &= \text{blue} \\
c(c) &= \text{green} \\
c(d) &= \text{blue} \\
c(e) &= \text{red}
\end{aligned}$$

Mutators: Vertex Removal

- Removes one or more vertices (and their incident edges).



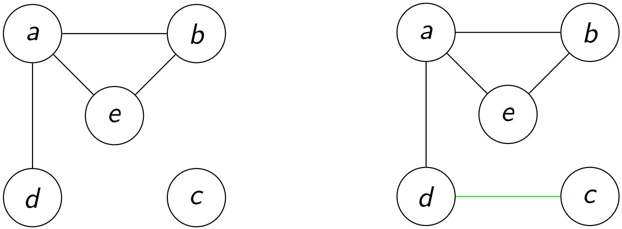
G



$G - \{c, e\}$

Mutators: Edge Addition

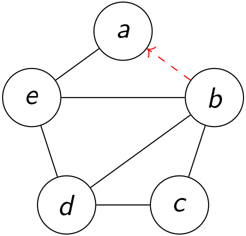
- Adds an edge between two non-adjacent vertices.
- Vertices are forced to have different colors in a proper coloring.



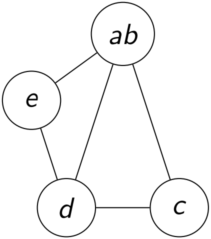
G $\quad$ $G + cd$

Mutators: Vertex Contraction

- Two vertices are identified as one.
- Any edge between the two vertices is discarded.
- Resulting multiple edges are reduced to a single edge.



G



$G \cdot ab$

Mutators: Complement

- Adjacent vertices in G are not adjacent in $\bar{G}$, and vice versa:

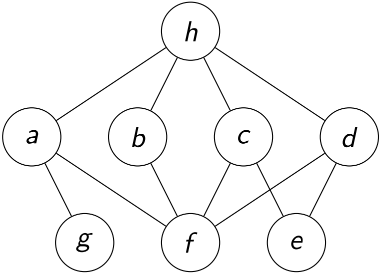
$$uv \in E(G) \iff uv \notin E(\bar{G})$$

G $\bar{G}$

Independent (Stable) Sets

- A subset of $V(G)$ whose elements are nonadjacent vertices.
- Maximal (MIS) if not a proper subset of some other independent set.
- Maximum if cardinality is $\geq$ any other MIS.
- The *independence number* $\alpha(G)$ is the cardinality of a maximum MIS in G.
- A proper coloring distributes vertices into independent sets.
- A chromatic coloring partitions vertices into independent sets.

MIS Example



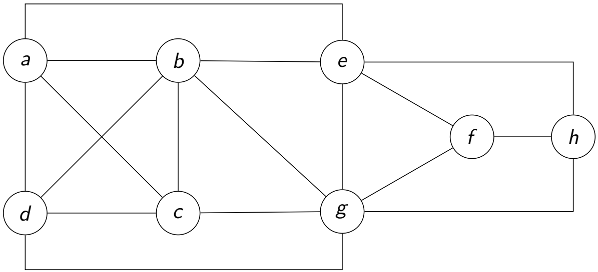
MIS	SIZE
$\{a, b, c, d\}$	4
$\{a, b, e\}$	3
$\{b, c, d, g\}$	4
$\{b, e, g\}$	3
$\{e, f, g, h\}$	4

$$\alpha(G) = 4$$

Cliques

- A complete graph embedded in (a subgraph of) a graph.
- A clique of order k is called a k-clique.
- A proper coloring for a graph with a k-clique requires at least k colors.
- Maximal if not a subgraph some other clique.
- Maximum if order is $\geq$ any other clique.
- The *clique number* $\omega(G)$ is the order of a max clique in G.
- A (maximal) clique in G is a (maximal) independent set in $\bar{G}$.
- $\omega(G) \leq \chi(G)$

Maximal Clique Example



G

MAXIMAL CLIQUE	ORDER
$G[\{a, b, c, d\}]$	4
$G[\{a, b, e\}]$	3
$G[\{b, c, d, g\}]$	4
$G[\{b, e, g\}]$	3
$G[\{e, f, g, h\}]$	4

$$\omega(G) = 4$$

Vertex Degree

- The *neighborhood* of a vertex $u \in V(G)$ is the set of all its neighbors:

$$N(u) = \{v \in V(G) \mid uv \in E(G)\}$$

- The *degree* of u is the cardinality of its neighborhood:

$$\deg(u) = |N(u)|$$

- First Theorem of Graph Theory:

$$\sum_{v \in V(G)} \deg(v) = 2m$$

Min/Max Degree

- Minimum degree:

$$\delta(G) = \min_{v \in V(G)} \deg(v)$$

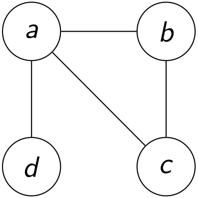
- Maximum degree:

$$\Delta(G) = \max_{v \in V(G)} \deg(v)$$

- Possible values:

$$0 \leq \delta(G) \leq \deg(v) \leq \Delta(G) \leq n - 1$$

Degree Example



G

$$\deg(a) = 3$$
$$\deg(b) = 2$$
$$\deg(c) = 2$$
$$\deg(d) = 1$$

$$\delta(G) = 1$$
$$\Delta(G) = 3$$

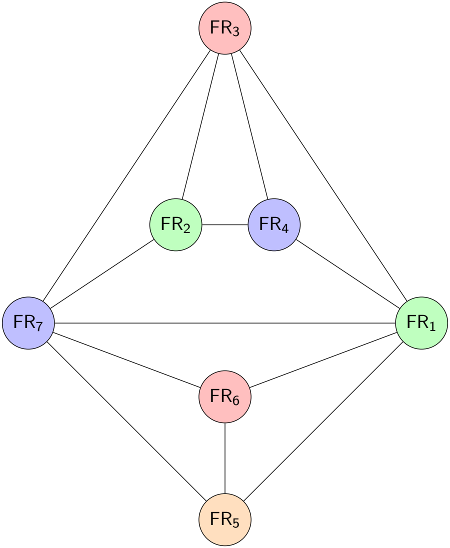
$$3 + 2 + 2 + 1 = 8 = 2 \cdot 4$$

Case Study: Toaster Design

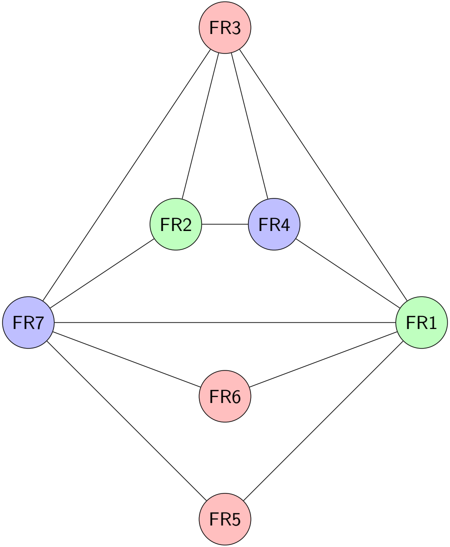
Functional Requirements

FR_1	Body contains all parts
FR_2	Can be safely moved while hot
FR_3	Can hold two slices of bread
FR_4	Heats each slice of bread on both sides
FR_5	Toasting is manually started
FR_6	Toasting is automatically or can be manually stopped
FR_7	Heat level can be controlled

Design 1: Four Parts



Design 2: Three Parts



Comparing Algorithms

- Methods:
  - Runtime Complexity (number of states)
  - Space Complexity (required memory)
  - Time Duration (execution time)
- Can be stated as best case, average case, and worst case.
- Best use is worst case in a specific problem domain.

Runtime Complexity

- Based on a length parameter of the problem: graph order n.
- Measured by Big-$\mathcal{O}$ notation: $\mathcal{O}(f(n))$ means the number of steps required to find a solution is $N \leq cf(n)$ for some $c > 0$.
- Most useful algorithms are P(olynomial)-time: $\mathcal{O}(n^c)$ for some $c \geq 0$.
- Inherently intractable algorithms are exponential (or worse) time: $\mathcal{O}(c^n)$ for some $c > 1$.
- Really meant to show asymptotic behavior at very large n.
- $\mathcal{O}(0.001n^2)$ and $\mathcal{O}(1000n^2)$ are still $\mathcal{O}(n^2)$.
- At lower n, P-time steps intended to "speed up" an algorithm can get in the way.
- Better to use empirical measurements for smaller n.

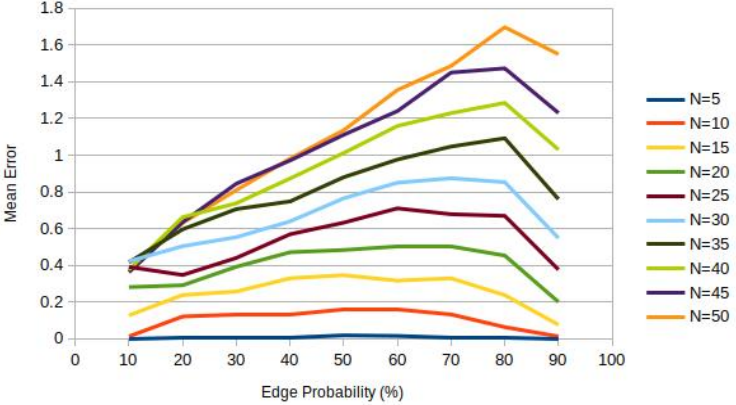
Random Graph Analysis

- Binomial edge probability model.
- Edge probabilities from 10% to 90% in steps of 10%.
- P-time algorithms:
  - $n = 5$ to $n = 50$ in steps of 1.
  - 1000 trials for each n.
- Exponential algorithms:
  - $n = 5$ to $n = 30$ in steps of 1.
  - 1000 trials for each $n < 20$.
  - 100 trials for each $n \geq 20$.

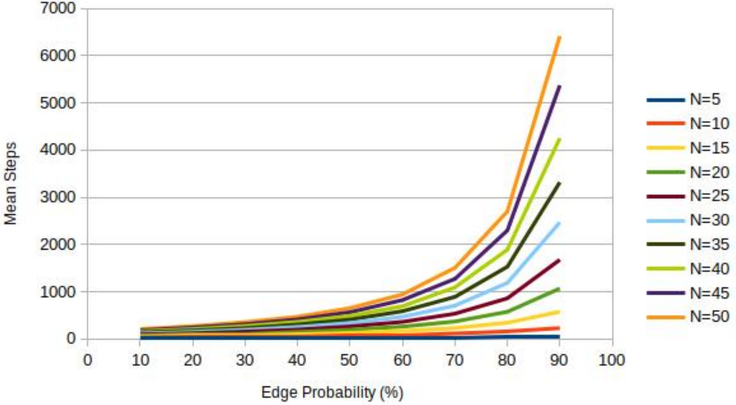
Estimating a Lower Bound

- $\omega(G) \leq \alpha(G)$.
- The clique number problem is also inherently intractable.
- Use the Edwards Elphick (1982) algorithm to find a lower bound: $\omega'(G) \leq \omega(G) \leq \chi(G)$.
- Label the vertices from 1 to n.
- Find a vertex v with $\deg(v) = \Delta(G)$.
- Add unselected vertices with lowest index that are adjacent to all selected vertices.
- Modification: choose the next vertex of highest degree (more time but increased accuracy).

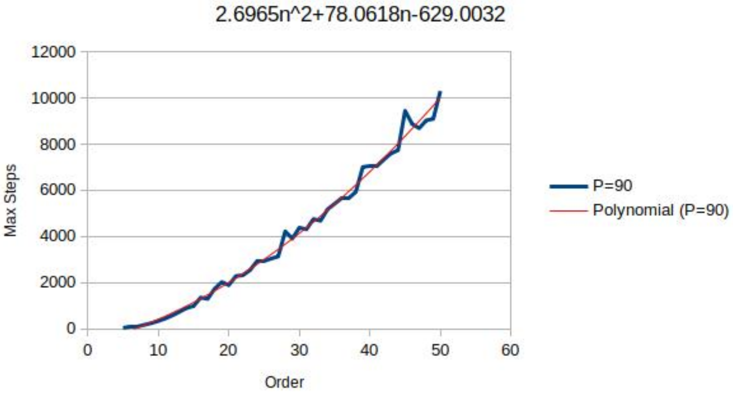
Improved Edwards Elphick Mean Error



Improved Edwards Elphick Mean Steps



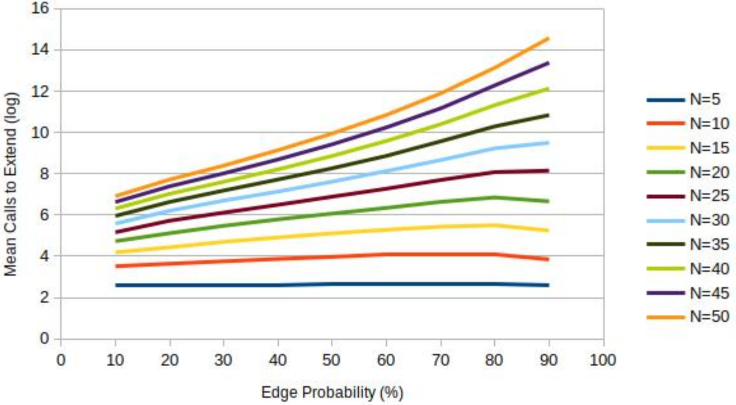
Improved Edwards Elphick Runtime Complexity



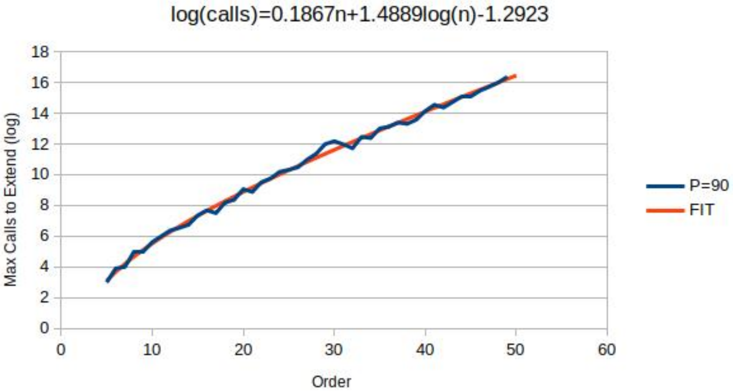
Bron Kerbosh Algorithm (1973)

- An exponential yet relatively fast algorithm for finding all maximal cliques in a graph.
- Moon and Moser (1965) show that $3^{\frac{n}{3}}$ is an upper bound.
- So the runtime complexity should be about $\mathcal{O}(1.44^n)$.
- Can be used to find MISs in $\bar{G}$.
- Thus can be used to determine $\omega(G)$ and $\alpha(G)$ exactly.
- Uses a recursive "extend" routine to extend the currently constructed clique.

Bron Kerbosh Mean Calls to Extend



Bron Kerbosh Runtime Complexity

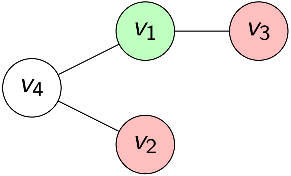


$$\mathcal{O}(2^{0.1867n}) = \mathcal{O}(1.14^n)$$

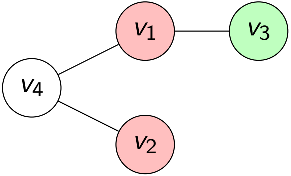
Estimating an Upper Bound

- Use a P-time sequential (greedy) algorithm that adds a new color when needed.
- Vertices are arranged and colored in a particular order.
- Matula (1967) concludes that ordering by non-increasing degree works best: last-first.
- Suggests color interchange to increase accuracy at the cost of greater complexity.
- Advantage: an empirical k-coloring.

Color Interchange

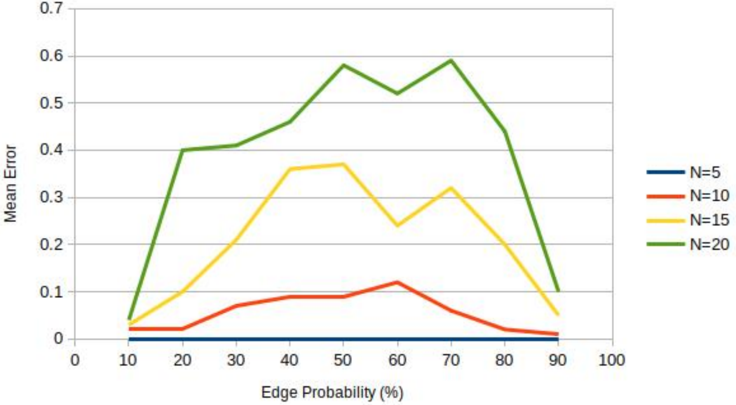


Before Swap

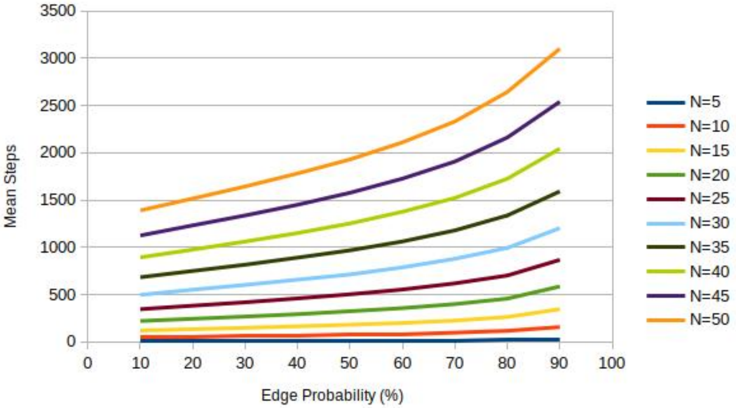


After Swap

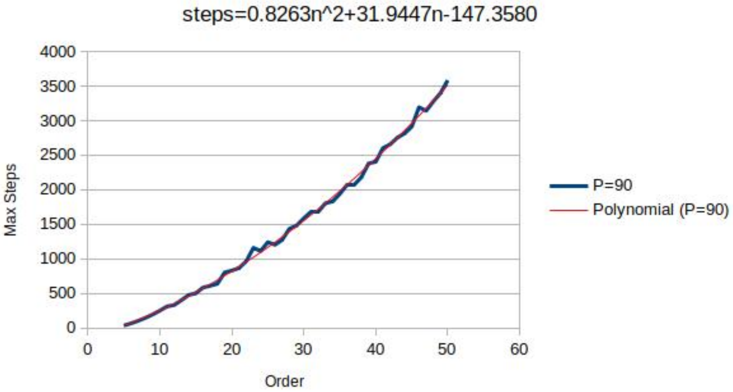
Last-First Greedy with Color Interchange Mean Error



Last-First Greedy with Color Interchange Mean Steps



Last-First Greedy with Color Interchange Runtime Complexity



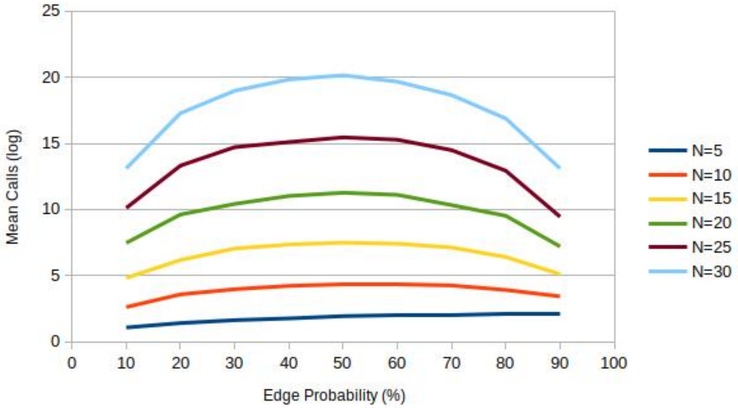
Christofides Algorithm (1971)

- An exponential algorithm to find an exact value for $\chi(G)$.
- Uses BK on $\bar{G}$ to find all MISs.
- Removes each MIS and recursively uses BK to find all MISs in what is left.
- Pieces the various ISs together to find the first combination that uses all the vertices.
- Includes processing to suppress IS combinations that will lead to duplicate results.

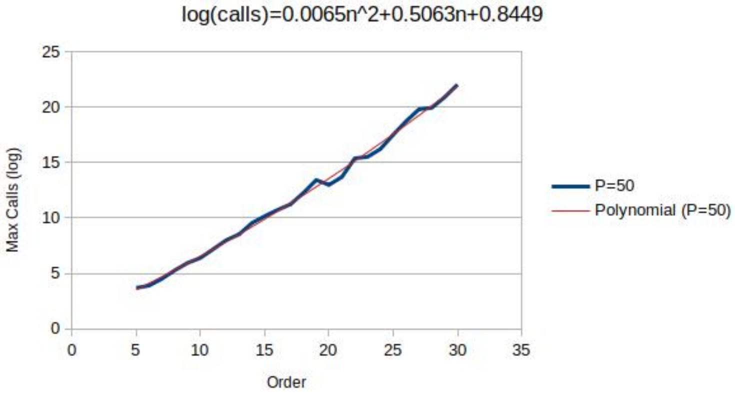
Wang Improvements (1974)

- Select a vertex $v \in V(G)$.
- Note that there is some chromatic coloring of G such that v is in some MIS of G.
- Can prove by starting with a chromatic coloring and stealing nonadjacent vertices from other ISs until the target IS is maximal.
- So choose a vertex that occurs in the least number of MISs and only consider those.

Christofides/Wang Mean Recursive Calls



Christofides/Wang Runtime Complexity



$$\mathcal{O}(2^{0.0065n^2}) = \mathcal{O}(1.0045^{n^2})$$

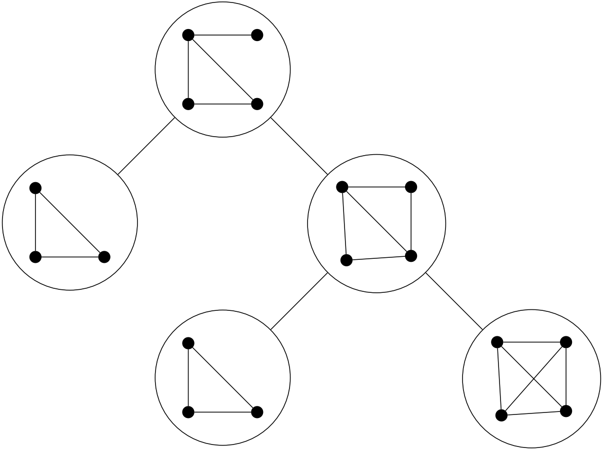
Zykov Algorithm (1949)

- An exponential algorithm to find an exact value for $\chi(G)$.
- For two nonadjacent $u, v \in V(G)$:

$$\chi(G) = \min\{\chi(G \cdot uv), \chi(G + uv)\}$$

- Vertex contraction forces u and v to have the same color.
- Edge addition forces u and v to have different colors.
- Recursive application exhaustively checks all possible combinations.

Zykov Algorithm Example



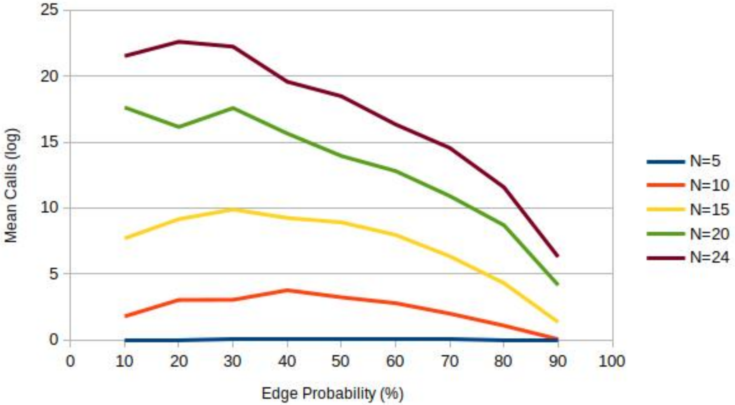
Branch and Bound

- The tree that tracks the states is called a Zykov tree.
- Each branch of the tree results from a vertex contraction (same color) or edge addition (different colors) decision between a pair of nonadjacent nodes.
- Certain bounding conditions are checked to see if a branch can be pruned.
- Otherwise, the termination condition for each branch is a complete graph.
- Each vertex in the complete graphs represents an independent set composed of contracted nodes.
- The complete graph with the smallest order is the chromatic number of the graph.

Zykov Bounding

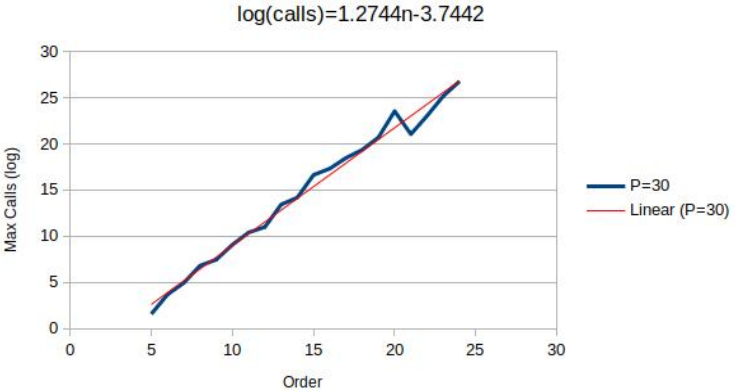
- Run a greedy algorithm once to establish an upper bound b.
- If a state graph has order less than b then set $b = n$.
- For each state, run Edwards Elphick to determine a lower bound.
- If the lower bound exceeds the current upper bound then prune.

Zykov Mean Recursive Calls



;

Zykov Runtime Complexity



;

$$\mathcal{O}(2^{1.7244n}) = \mathcal{O}(3.3^n)$$

Christofides/Wang vs Zykov

- CW: $\mathcal{O}(1.0045^{n^2})$; Z: $\mathcal{O}(3.3^n)$
- CW executes the exponential BK at every state.
- Z per-state processing is very light.
- CW is better at pruning states in the target range.
- CW is faster in the target range.
- CW and Z number of states become equal at $n = 112$.
- Due to less per-state processing, Z should overtake CW before that.

Proposed Algorithm

- Can we do better than Christofides by modifying Zykov?
- Try looping on increasing values of k.
- Determine if a particular graph state is k colorable.
- Introduce more bounding conditions based on the current k.
- Prune graphs that fail the bounding conditions.
- Introduce mutations that simplify state graphs with equivalent colorability.
- Introduce success conditions that indicate when a state graph is k-colorable.
- First successful k wins.

Proposed Algorithm Outline

- Run BK to determine k_{min}.
- Run greedy to determine k_{max} and a candidate coloring.
- Use BK to determine a set of candidate root graphs with the vertices in each target MIS contracted.
- Loop on k from k_{min} to k_{max}.
- For each k value and each candidate tree, run a modified Zykov algorithm to determine if the candidate graph is k-colorable.
- If k_{max} is achieved then accept the greedy candidate.

Modified Zykov Routine

- Success condition: $n \leq k$.
- Maximum edge threshold:

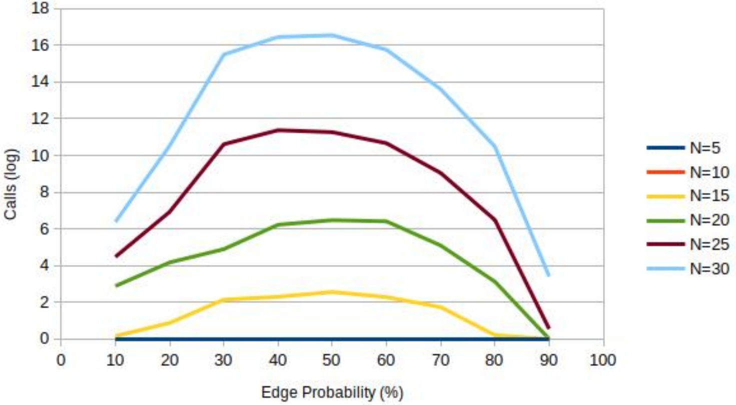
$$m \leq \frac{n^2(k-1)}{2k}$$

- Remove vertices with degree $< k$.
- If $N(u) \subset N(v)$ then contract.
- Minimum common neighbors upper bound:

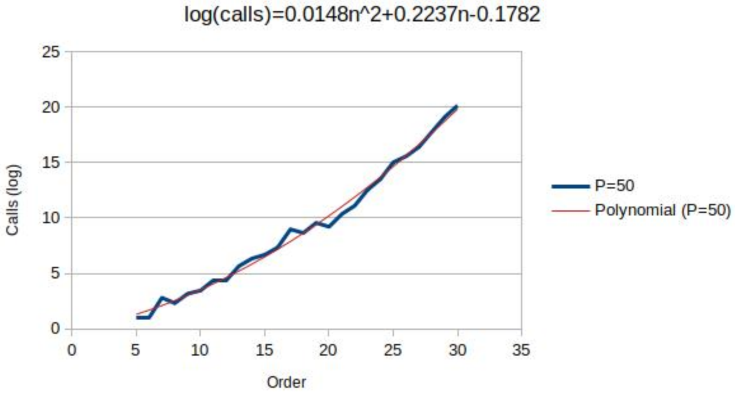
$$\min_{u,v \in V(G)} |N(u) \cap N(v)| \leq n - 2 - \frac{n-2}{k-1}$$

- Edwards Elphick to calculate ω' and then $w' \leq k$.
- Otherwise branch.

Proposed Algorithm Mean Calls

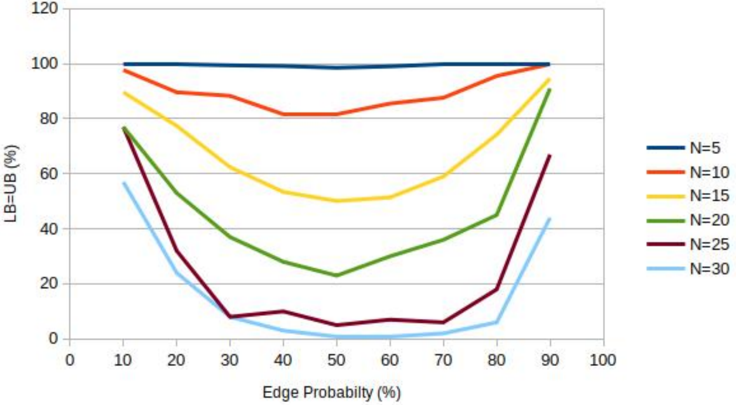


Proposed Algorithm Runtime Complexity

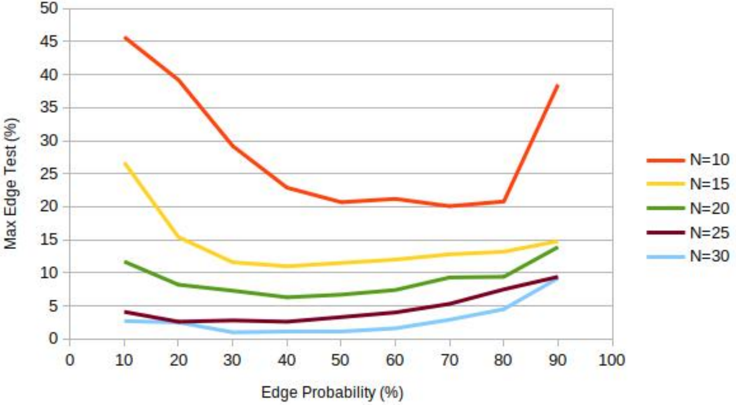


$$\mathcal{O}(2^{0.0148^2}) = \mathcal{O}(1.0103^{n^2})$$

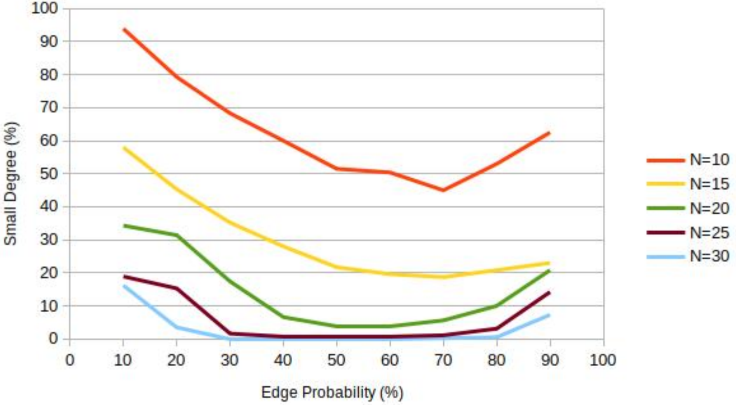
Step Effectiveness: Success Condition



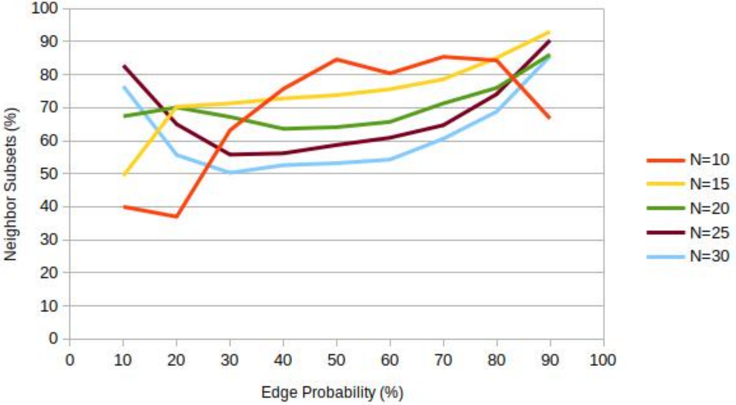
Step Effectiveness: Maximum Edge Threshold



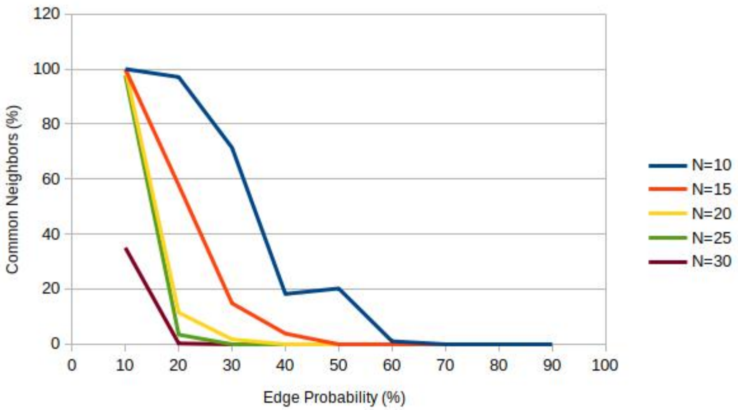
Step Effectiveness: Small Vertex Degree Removal



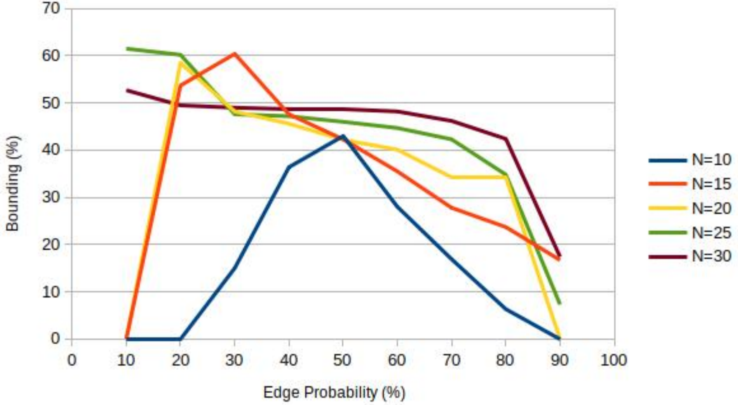
Step Effectiveness: Neighborhood Subset



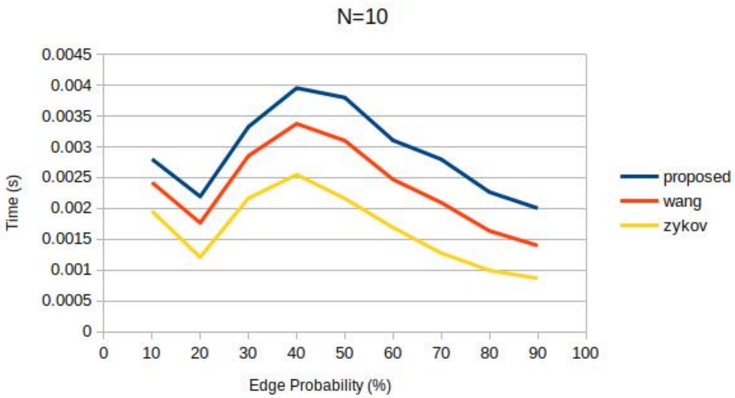
Step Effectiveness: Minimum Common Neighbors Upper Bound



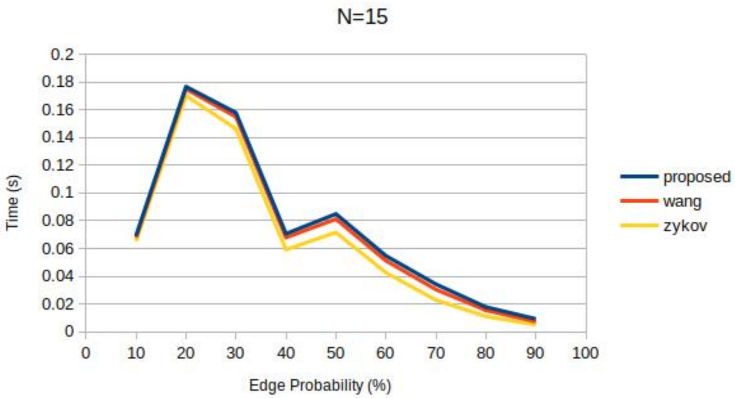
Step Effectiveness: Standard Zykov Bounding



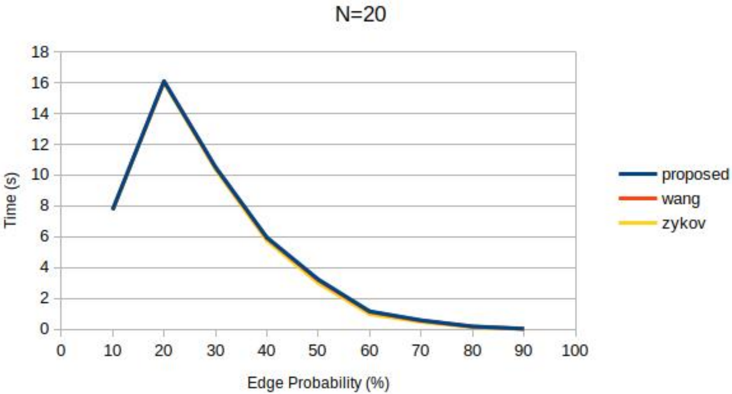
Time Duration: $n = 10$



Time Duration: $n = 15$



Time Duration: $n = 20$



Conclusions

- Any of the three algorithms are sufficient for the design tool in the target range.
- Bounding conditions influence runtime complexity at small to moderate n.
- Most bounding conditions effectively die out with at larger n.
- Intractable problems must be broken up into problem domains.
- There is no one solution that works in all cases.