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

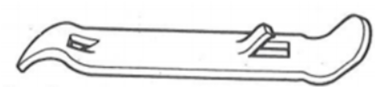
An optimal design always maintains the independence of the FRs. This means that the FRs and DPs are related in such a way that a specific DP can be adjusted to satisfy its corresponding FR without affecting other FRs.

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.

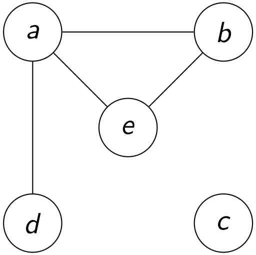
The Chromatic Coloring Problem

- 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.

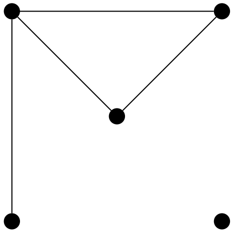
Simple Graphs

- 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: $m = |E(G)|$.
- $m \leq \binom{n}{2} = \frac{n(n-1)}{2}$

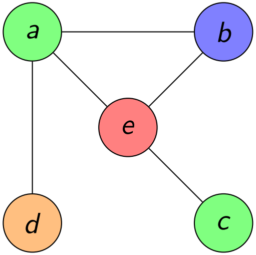
Order/Size Special Cases

- 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: $\chi(G)$.

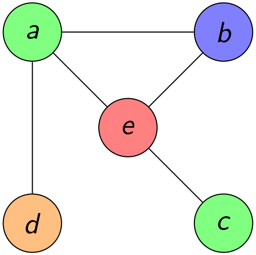
4-coloring Example



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

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

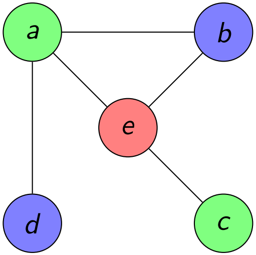
5-coloring Example



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

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

Chromatic Coloring Example

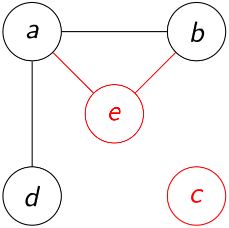


$$C = \{\text{green}, \text{blue}, \text{red}\}$$

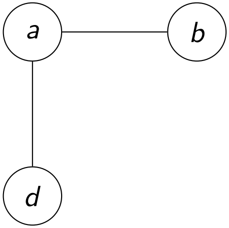
$$\begin{aligned} c(a) &= \text{green} \\ c(b) &= \text{blue} \\ c(c) &= \text{green} \\ c(d) &= \text{blue} \\ c(e) &= \text{red} \end{aligned}$$

Mutators: Vertex Removal

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



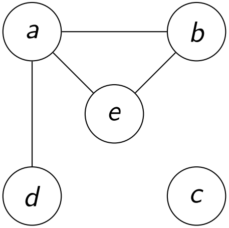
G



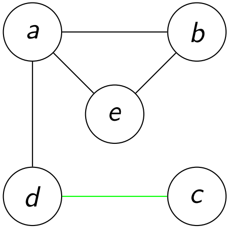
$G - \{c, e\}$

Mutators: Edge Addition

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



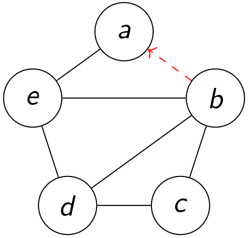
G



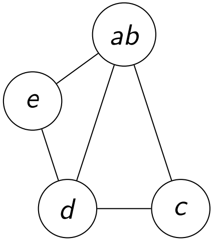
$G + cd$

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

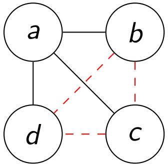


$G \cdot ab$

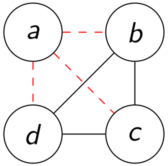
Mutators: Complement

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

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



G

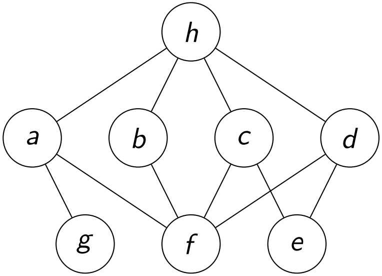


$\bar{G}$

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



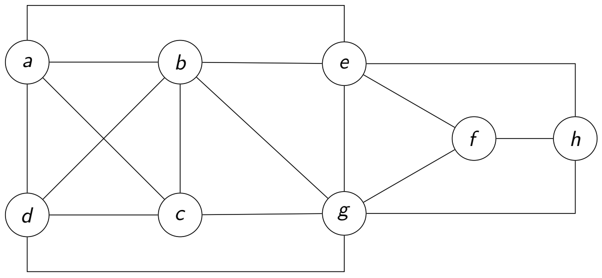
MIS	SIZE
$\{a, b, c, d\}$	4
$\{a, b, e\}$	3
$\{b, c, d, g\}$	4
$\{b, e, g\}$	3
$\{e, f, g, h\}$	4

$$\alpha(G) = 4$$

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:

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

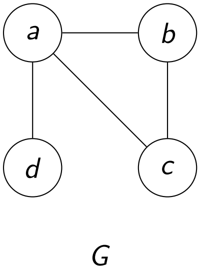
- Maximum degree:

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

- Possible values:

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

Degree Example



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

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

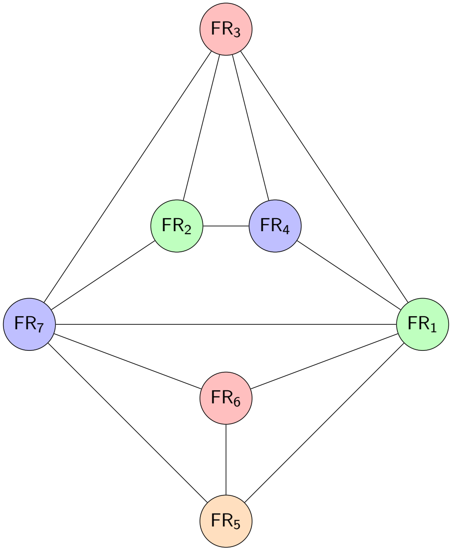
$$3 + 2 + 2 + 1 = 8 = 2 \cdot 4$$

Case Study: Toaster Design

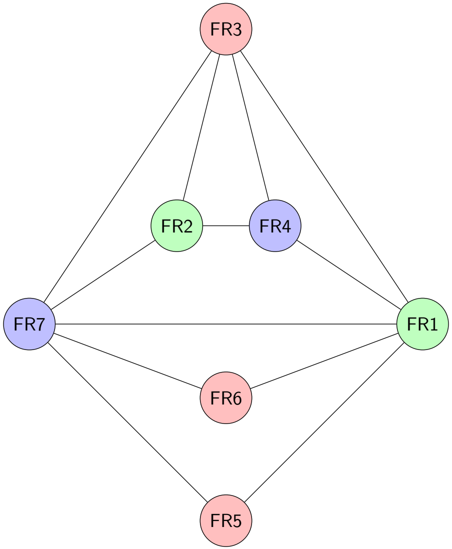
Functional Requirements

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.

Random Graph Analysis

- Binomial edge probability model.
- Edge probabilitys 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 $n \geq 20$.