Proceedings of the ASME 2019 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2019
August 18–21, 2019, Anaheim, California, USA

IDETC2019-98251

A GRAPH COLORING TECHNIQUE FOR IDENTIFYING THE MINIMUM NUMBER OF PARTS FOR PHYSICAL INTEGRATION IN PRODUCT DESIGN

Praveen Kumare Gopalakrishnan
Graduate Research Assistant
Department of Mechanical and Aerospace Engineering
Industrial and Systems Engineering Department
University at Buffalo, SUNY
Buffalo, NY 14260
Email: pgopalak@buffalo.edu

Jeffery Cavallaro
Graduate Research Assistant
Department of Mathematics and Statistics
San Jose State University
San Jose, CA 95192
Email: jeffery.cavallaro@sjsu.edu

Sogol Jahanbekam
Assistant Professor
Department of Mathematics and Statistics
San Jose State University
San Jose, CA 95192
Email: sogol.jahanbekam@sjsu.edu

Sara Behdad
Assistant Professor
Department of Mechanical and Aerospace Engineering
Industrial and Systems Engineering Department
University at Buffalo, SUNY
Buffalo, NY, 14260
Email: sarabehd@buffalo.edu

ABSTRACT

The objective of this study is to develop a mathematical framework for determining the minimum number of parts required in a product to satisfy a list of functional requirements (FRs) given a set of connections between FRs. The problem is modeled as a graph coloring technique in which a graph G with n nodes (representing the FRs) and m edges (representing the connections between the FRs) is studied to determine the graph's chromatic number $\chi(G)$, which is the minimum number of colors required to properly color the graph. The chromatic number of the graph represents the minimum number of parts needed to satisfy the list of FRs. In addition, the study calculates the computational efficiency of the proposed algorithm. Several examples are provided to show the application of the proposed algorithm.

NOMENCLATURE

FR A functional requirement in a design
DP A design parameter
(FR) A vector of functional requirements
(DP) A vector of design parameters
(A) A square, full-rank design matrix that relates FRs to DPs by the matrix equation $(FR) = (A)(DP)$
G A graph, which is a mathematic object consisting of a set of nodes and a set of edges
$V(G)$ The set of all nodes in graph G
$E(G)$ The set of all edges in graph G
$n(G)$ The order of (number of nodes in) graph G
$m(G)$ The size of (number of edges in) graph G
$u, v, w, \ldots$ Nodes in a graph
uv An edge in a graph between nodes u and v
$N(v)$ The neighborhood of node v
$\mathrm{d}(v)$ The number of edges incident to node v in a graph
c A coloring of a graph
$G - v$ The graph formed from G with node v and all of its inci-

dent edges removed

$G-X$ The graph formed from G with nodes in some set $X \subseteq V(G)$ and all of their incident edges removed

$G \cdot uv$ The graph formed from G by contracting nodes u and v

$G+uv$ The graph formed from G by adding edge uv

$\chi(G)$ The chromatic number of graph G

BACKGROUND

The objective of this study is to determine the minimum number of parts required in a product to satisfy the list of FRs. Therefore, in this section, we provide an overview of the concepts of physical integration, part consolidation, part integration, modular design, and part clustering and how they have been presented in the literature as a means to reduce the number of parts in a product.

Physical Integration

The concept of physical integration was first introduced in the Axiomatic Design field by Professor Nam P. Suh at MIT in 1976. Per Axiomatic Design, a good design should satisfy two main axioms: the Independence Axiom and the Information Axiom. The Independence Axiom states that the functional requirements (FRs) of a design should be independently satisfied by a list of Design Parameters (DPs). The Information Axiom states that a design with lower information content or lower complexity has a higher probability of success [1]. In addition, Professor Suh believes that a simple design — a design in which the list of FRs can be satisfied by a limited number of parts — is a good design. This suggests the idea of physical integration in order to achieve a lower number of parts in a product.

It should be noted that having a lower number of parts does not conflict with the Independence Axiom. The statement of FR independence is embodied in the design matrix equation:

$$(FR) = (A)(DP) \tag{1}$$

where (FR) is a vector of some number of functional requirements, (DP) is a vector of an equal number of design parameters, and (A) is a square, full-rank design matrix. Thus, a higher number of FRs (rows of (A)) requires an equally higher number of DPs (columns of (A)) in order to maintain the independence of the FRs, but does not necessarily require a higher number of parts [2].

Concepts such as part consolidation in additive manufacturing [3–5], modular design, and part clustering convey a similar message as physical integration, since all of them aim to reduce the number of parts or modules within a product. However, physical integration is beyond just part consolidation or improving the degree of modularity in a design, as it aims to increase the degree of physical integration while simultaneously satisfying the independence of the FRs. The number of studies that have addressed the concept of physical integration is very limited, therefore we briefly provide a review of studies that have covered the concepts of part consolidation, modular design, and part clustering.

Part Consolidation

Part consolidation is a capability offered by additive manufacturing that makes the design and manufacturing of complex products possible. It can be used with the aim of reducing the weight and height of a multipart assembly [4]. Yang et al. [5] developed a part consolidation method that uses FRs, an initial CAD model, and performance requirements as inputs to first integrate feasible functions and then optimize the structure of the design. In another study [3], the same group attempted to automate the traditional procedure for applying design heuristics that are used to select potential candidates for part consolidation. They formulated the problem as a graph and developed an algorithm to group parts as part consolidation candidates considering the physical attributes of the parts. Tang et al. [6] commented that functionality integration and part consolidation will improve the sustainability of the design by reducing part count and improving performance.

Modular Design

The modular design concept divides the design into smaller modules that can be independently analyzed or can be used interchangeably with different modules [7]. Modular design uses pre-defined modular interfaces, industry standard interfaces for functional partitioning of reusable modules, and discrete scalability [8]. These concepts use modules that are similar to the FRs used in Axiomatic Design. The FRs are independent on their own and have their own function just like modules and can be used for different designs or in other words combined with different modules. Applications vary from biomedical (protein analysis), where similar structured proteins are considered as modules and their interactions are studied [9–11], to product design studies focused on product family design [12–14]. These researchers show how the modular design concept uses modules as independent entities used for clustering analysis.

Part Clustering

Graph theory and network analysis is a promising approach for clustering analysis and decision-making in engineering design [15–17]. Converting design problems to graphs, or their equivalent matrices, helps designers understand them better as abstractions of real-world problems. There are different ways of analyzing the graphs. Graph coloring is a preferred technique [18] that offers a wide range of applications like graph partitioning algorithms, graph clustering techniques, ranking of

graphs, and complexity analysis [19].

The graph partitioning approach has already been used for solving design problems. Li and his group worked on developing a knowledge-based graph partitioning technique to identify reusable CAD model designs [20]. Wolfie et. al. developed a graph partitioning technique for watermarking in VLSI design [21, 22]. Other design problems such as topological design for industrial networks [23], RNA graph partitioning [24], SMART partitioning to analyze datasets [25], and graph partitioning for multi-processor system-on-a-chip design [26] are all examples of successful implementations of graph partitioning techniques. Similar to partitioning techniques, graph clustering has also been widely used for analyzing design optimization problems [27], data analysis [28], and structural design [29].

While graph partitioning techniques have already been used in the design domain for different purposes, the application of them for defining the number of parts is quite new. The main contribution of this study is to develop a new graph partitioning technique for identifying the number of parts within a product by considering the compatibility of FRs.

GRAPH THEORY BASICS

In order to better understand the proposed algorithm, a short introduction to the concepts and terminology of graph theory is provided in this section.

Nodes and Edges

A *graph* G is a mathematical object consisting of two sets: a set of *nodes* (*vertices*), denoted by $V(G)$; and a set of *edges*, denoted by $E(G)$. Each edge in $E(G)$ is associated with two (possibly the same) nodes in $V(G)$, referred to as the edge's *endpoints*. The endpoints of an edge are said to be *adjacent*. An edge and its endpoints are said to be *incident*. A node with no incident edges is said to be *isolated*.

For our application, it is sufficient to use a restricted set of graphs called *simple* graphs; these are graphs with no *loop* edges on a single node and no *multiple* edges between two nodes. Thus, an edge's endpoints are always distinct and each pair of nodes has at most one incident edge.

Individual nodes in a graph are typically identified by lowercase letters: $u, v, w, \ldots \in V(G)$. A node can also be identified by a descriptive *label*. Edges are typically identified by juxtaposition of their endpoints: $uv \in E(G)$. Note that simple graphs are not directed, so uv and vu denote the same edge.

Order and Size

The *order* of a graph G, denoted by $n(G)$ or just n if the graph in question is unambiguous, is the number of nodes in G:

$$n = n(G) = |V(G)| \tag{2}$$

A graph with no nodes (and hence no edges) is called the *null* graph.

The *size* of a graph G, denoted by $m(G)$ or just m if the graph in question is unambiguous, is the number of edges in G:

$$m = m(G) = |E(G)| \tag{3}$$

A graph with no edges is called *trivial*.

The maximum number of edges in a graph G is given by the following theorem:

Theorem 1. *Let G be a graph of order n and size m:*

$$m \leq \frac{n(n-1)}{2} \tag{4}$$

A graph with the maximum number of edges (hence all of the nodes in G are adjacent with each other) is called *complete*.

Neighborhood and Degree

Given a graph G and a node $v \in V(G)$, the *neighborhood* of v, denoted by $N(v)$, is the set of all nodes in G that are adjacent to v:

$$N(v) = \{u \in V(G) \,|\, uv \in E(G)\} \tag{5}$$

The *degree* of v, denoted by $\mathrm{d}(v)$, is the number of edges in G that are incident to v, which corresponds to the number of nodes in G that are adjacent to v. In other words:

$$\mathrm{d}(v) = |N(v)| \tag{6}$$

For a graph of order n, it is the case that $0 \leq d(v) < n$. Note that for an isolated node v: $N(v) = \emptyset$ and $d(v) = 0$.

The degrees of the nodes in a graph and the size of the graph are related by the so-called *Fundamental Theorem of Graph Theory*:

Theorem 2 (Fundamental Theorem of Graph Theory). *Let G be a graph of size m:*

$$\sum_{v \in V(G)} \mathrm{d}(v) = 2m \tag{7}$$

Coloring

A *coloring* c of a graph G is a function $c : V(G) \to C$, where C is a set of "colors." Thus, the color of a node is nothing more

than an attribute of the node. Although the elements of C are usually actual colors (red, green, blue, etc.), a graph coloring problem is free to select any value type for the color attribute. Note that there is no assumption that c is surjective, so the codomain C may contain unused colors.

A coloring c on a graph G is called *proper* when:

$$\forall u, v \in V(G), v \in N(u) \implies c(u) \neq c(v) \tag{8}$$

In other words, no two adjacent nodes have the same color.

A proper coloring c of a graph G where $|C| = k$ is called a *k-coloring* of G; the coloring uses *at most* k colors. A graph that has a k-coloring is called *k-colorable*. Since there is no requirement to use all of the colors in a k-coloring of a graph G, we can make the following statement:

Proposition 1. *Let G be a graph:*

$$G \text{ is } k\text{-colorable} \implies G \text{ is } (k+1)\text{-colorable}$$

The minimum k such that G is k-colorable is called the *chromatic number* of G, denoted by $\chi(G)$. A k-coloring for a graph G where $k = \chi(G)$ is called *k-chromatic*.

The primary purpose of a k-coloring of a graph G is to distribute the nodes of G into k so-called *independent* (some possibly empty) sets, where all of the nodes in an independent set are non-adjacent. We use the term "distribute" instead of the term "partition" since the formal definition of a partition does not allow for empty sets. However, note that when a coloring is chromatic, there are no empty sets and the distribution is a true partition.

PURPOSE

The purpose of this paper is to develop a graph coloring technique for determining the minimum number of parts required in a product to satisfy a list of functional requirements (FRs) considering a set of connections between FRs. It is assumed that the designer has already determined the FRs, which are represented by the nodes of a graph, and has determined which FRs cannot be satisfied in the same part, represented by edges between incompatible FR nodes. The solution then becomes determining the chromatic number for the resulting graph.

The development of FRs is driven by the customer attributes from the customer domain and the FRs are subject to the independence and information axioms [30]. Thus, the determination of the nodes in the graph tend to be relatively clear. The determination of the connections between the FRs may not be so straightforward. In practice, the compatibility of the nodes is defined by the likelihood that the designer would like to have certain FRs together. In fact, the graph is defined based on certain predefined feasibility principles used by designers. One possible framework for determining the edges in the graph may be found in the movement, material, and assembly/disassembly constraints of Boothroyd and Dewhurst's design for manufacture and assembly (DFMA) [31]. Regardless, FR and edge determination are assumed to have been decided prior to application of the proposed algorithm, and thus the methods for doing so are beyond the scope of this paper.

It is important to note that the goal of the proposed algorithm is to determine the chromatic number for the resulting graph, not to determine a particular chromatic coloring. Determining the chromatic number is an NP-hard problem. The typical approach in graph theory is to apply various theorems to try and determine a lower bound, and then use some heuristic algorithm like *greedy coloring* to determine an upper bound. If one gets lucky, the upper and lower bounds match and the chromatic number is thus found. All known exact algorithms run in exponential time. So, like most exponential-time algorithms, the goal is to develop an algorithm that shaves magnitude from the exponent.

In one of the previous papers of the authors [32], a graph coloring technique was defined to determine the number of parts; however, that algorithm was not computationally efficient and the user was required to test the algorithm for a list of a given number of parts. In this paper, we aim to overcome the limitations of the previous algorithm and prove a theorem to determine the number of parts in a closed-form computational timeframe. Moreover, the proposed algorithm considers the absolute compatibility of FRs unlike the previous algorithm. which integrates the weighted connections between FRs. Overall, while both algorithms have the same objective of minimizing the number of parts, their applications and the scopes of their usage would be different.

PROPOSED ALGORITHM

We start with a precise statement of the proposed algorithm. This is followed by a more detailed description of the algorithm's steps and the application of those steps to an example graph. Finally, for the hearty reader, the theoretical basis for the algorithm and the algorithm's computational complexity is presented.

The input to the algorithm is a graph G whose nodes represent the functional requirements (FRs) of a product and whose edges represent the connections between those FRs: adjacent FRs cannot be combined into a single part due to some design constraint enforced by the designer. The output is $\chi(G)$, which represents the minimum number of parts required to satisfy all of the FRs in the product. Once this number is known, any solution from a heuristic algorithm such as the greedy coloring algorithm that uses this number of colors in known to be ideal. FRs of the same color can then be combined into the same part.

The algorithm is divided into two parts: a recursive subroutine and an outer main loop.

Recursive Subroutine

The recursive subroutine: is-k-colorable(G,k), is responsible for determining whether the specified graph G is k-colorable for a specified value of k. It returns either true or false. Any alterations that the subroutine makes to G are passed back to the calling main loop.

The steps of the recursive subroutine are as follows:

1. If $n \leq k$ then return true (Proposition 2).
2. If $m > \frac{n}{2k}(kn-n)$ then return false (Lemma 1).
3. Let $X = \{v \in V(G) | \, \mathrm{d}(v) < k\}$. If $|X| = 0$ then go to 4. Otherwise, replace G with $G - X$ and go to 1 (Corollary 2).
4. If G has nodes u and v such that $N(u) \subseteq N(v)$ then replace G with $G - u$ and go to 1 (Lemma 4).
5. Select two non-adjacent nodes $u, v \in V(G)$ with the smallest number of common neighbors. If $k \geq 2$ and $|N(u) \cap N(v)| > n - 2 - \frac{n-2}{k-1}$ then return false (Corollary 3).
6. Return is-k-colorable($G \cdot uv$, k) or is-k-colorable($G + uv$, k) (Lemma 7).

The subroutine is guaranteed to return because either there will be sufficient node reductions and/or contractions such that $n \leq k$ or sufficient edge additions such that the graph becomes complete and $m > \frac{n}{2k}(kn-n)$ for any $k < n$.

Main Loop

The main loop: find-k-colorable(G), is responsible for determining the chromatic number for the specified graph G. It returns $\chi(G)$.

The steps of the main loop are as follows:

1. If $n = 0$ then return 0.
2. Set $k = 1$.
3. If is-k-colorable(G, k) then return k.
4. $k = k + 1$
5. Go to 3.

The main loop is guaranteed to complete because eventually k will exceed n, which will cause the called subroutine to return true.

ALGORITHM DESCRIPTION

The main loop of the algorithm tries increasing values of k until the called recursive subroutine confirms that G is k-colorable. This will find the smallest such k, which is indeed the desired $\chi(G)$ value. Thus, the rest of this section will focus on the recursive subroutine.

The recursive subroutine starts with one fairly straightforward fact: if $n \leq k$ then G must be k-colorable because each node can be assigned its own color:

Proposition 2. *Let G be a graph of order n and let $k \in \mathbb{N} \cup \{0\}$:*

$n \leq k \implies$ *G is k-colorable.*

Thus, the strategy of the recursive subroutine is to attempt to decompose G into progressively simpler graphs with fewer nodes such that G is k-colorable iff the simpler graphs are k-colorable. If such a simpler graph can be found of order $n \leq k$ then we can conclude that the original G is indeed k-colorable. Otherwise, the main loop increments k and we try again.

The steps of the recursive subroutine are described in the following sections.

STEP 1: Checking for Success

This step applies the success condition of Proposition 2. If $n \leq k$ then every node can have its own color and the graph is k-colorable so return true. Otherwise, continue with the next step.

STEP 2: Edge Density Test

As the number of edges increases the number of adjacencies increases thus requiring a larger k. This test establishes an edge density threshold:

$$\frac{n}{2k}(kn-n) \tag{9}$$

If the size m exceeds this threshold then the graph cannot be k-colorable so return false. This will cause the main loop to increment k and try again. Otherwise, continue with the next step.

STEP 3: Removing Low-degree Nodes

Nodes with degrees $< k$ are unlikely to affect k-colorability. In fact, they (and their incident edges) can be removed without affecting the k-colorability of a graph. This is the most efficient order-reducing step of the algorithm. If some nodes are removed then go to step 1. Otherwise, continue with the next step.

STEP 4: Removing Neighborhood Subsets

A node whose neighborhood is a subset of another node's neighborhood does not affect the overall colorability and can be removed (along with its incident edges). This is the second and slightly more complex order-reducing step of the algorithm. If such a node is found and removed then go to step 1. Otherwise, continue with the next step.

STEP 5: Smallest Neighborhood Intersection Test

If there are no more low-degree nodes or neighborhood subsets then there is one more check that can be applied before having to resort to expensive recursion. Two nodes with the smallest

neighborhood intersection are located and then the following calculation is made:

$$n-2-\frac{n-2}{k-1} \tag{10}$$

This is an upper bound for the smallest possible neighborhood intersection. If the intersection of the previously found nodes exceeds this value then the graph cannot be k-colorable so return false. This will cause the main loop to increment k and try again. If not, then proceed with the recursive step.

STEP 6: Recursive Step

It is the recursive steps that make algorithms like this so expensive. Hopefully, a determination for $\chi(G)$ is made by the preceding steps. If not, then recursive calling is necessary to further decompose the graph.

First, it is assumed that the graph is k-colorable with some proper coloring c. Now, consider the nodes determined in the previous smallest neighborhood step. If such a c exists, then the two nodes can have either the same color or different colors. The same color allows the two nodes to be compacted in order to reduce the order of the graph. Different colors allow an edge to be added between the two nodes. The goal of the extra edge is to affect the subsequent edge density calculation.

ALGORITHM EXAMPLE

Graphs are better visualized pictorially, where the nodes are drawn as circles with labels and edges are drawn as lines between adjacent nodes. Thus, the algorithm will be demonstrated using the sample graph in Figure 1.

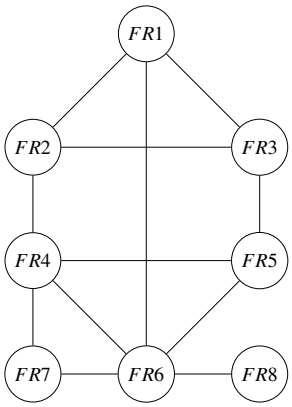


FIGURE 1. SAMPLE GRAPH

Note that for the sample graph: $n = 8$ and $m = 12$.

1. (main) Since $n = 8 > 0$, G is not the null graph so set $k = 1$ and call the recursive subroutine.
2. (sub-0) Since $n = 8 > 1 = k$, do not return.
3. (sub-0) Perform the edge density calculation for $n = 8, k = 1, m = 12$:

$$\frac{n}{2k}(kn-n) = \frac{8}{2\cdot 1}(1\cdot 8-8) = 0 < 12 = m \tag{11}$$

   Thus, G is not 1-colorable so return false.
4. (main) Set $k = 2$ and call the recursive subroutine.
5. (sub-0) Since $n = 8 > 2 = k$, do not return.
6. (sub-0) Perform the edge density calculation for $n = 8, k = 2, m = 12$:

$$\frac{n}{2k}(kn-n) = \frac{8}{2\cdot 2}(2\cdot 8-8) = 16 \geq 12 = m \tag{12}$$

   Thus, G may be 2-colorable so do not return.
7. (sub-0) Since $d(FR8) = 1 < 2 = k$, replace G with $G - \{FR8\}$ as shown in Figure 2.

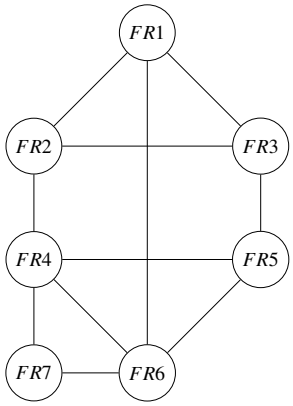


FIGURE 2. FR8 REMOVED

8. (sub-0) The new G now has $n = 7$ and $m = 11$. Since $n = 7 > 2 = k$, do not return.
9. (sub-0) Perform the edge density calculation for $n = 7, k = 2, m = 11$:

$$\frac{n}{2k}(kn-n) = \frac{7}{2\cdot 2}(2\cdot 7-7) = 12.25 > 11 = m \tag{13}$$

   Thus, G may be 2-colorable so do not return.
10. (sub-0) There are no nodes with degree < 2 so continue.

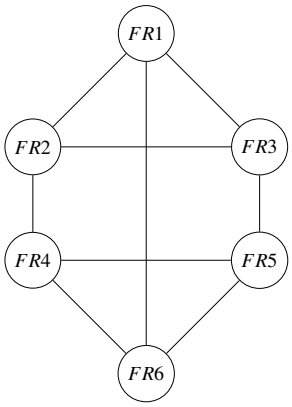


FIGURE 3. FR7 REMOVED

11. (sub-0) Since $N(FR7) \subseteq N(FR5)$, replace G with $G - FR7$ as shown in Figure 3.
12. (sub-0) The new G now has $n = 6$ and $m = 9$. Since $n = 6 > 2 = k$, do not return.
13. (sub-0) Perform the edge density calculation for $n = 6, k = 2, m = 9$:

$$\frac{n}{2k}(kn - n) = \frac{6}{2 \cdot 2}(2 \cdot 6 - 6) = 9 \geq 9 = m \qquad (14)$$

Thus, G may be 2-colorable so do not return.
14. (sub-0) There are no nodes with degree < 2 so continue.
15. (sub-0) There are no neighborhood subsets so continue.
16. (sub-0) By the symmetry of the graph, we can see that any two non-adjacent nodes have two common neighbors. So select $FR1$ and $FR4$ and perform the minimum neighborhood intersection check:

$$n - 2 - \frac{n-2}{k-1} = 6 - 2 - \frac{6-2}{2-1} = 0 < 2 \qquad (15)$$

Thus, G is not 2-colorable so return false.
17. (main) Set $k = 3$ and call the recursive subroutine.
18. (sub-0) Since $n = 6 > 3 = k$, do not return.
19. (sub-0) Perform the edge density calculation for $n = 6, k = 3, m = 9$:

$$\frac{n}{2k}(kn - n) = \frac{6}{2 \cdot 3}(3 \cdot 6 - 6) = 12 \geq 9 = m \qquad (16)$$

Thus, G may be 3-colorable so do not return.
20. (sub-0) There are no nodes with degree < 2 so continue.
21. (sub-0) There are no neighborhood subsets so continue.
22. (sub-0) By the symmetry of the graph, we can see that any two non-adjacent nodes have two common neighbors. So select $FR1$ and $FR4$ and perform the minimum neighborhood intersection check:

$$n - 2 - \frac{n-2}{k-1} = 6 - 2 - \frac{6-2}{3-1} = 2 \geq 2 \qquad (17)$$

Thus, G may be 3-colorable so do not return.
23. (sub-0) Contract $FR1$ and $FR4$ as shown in Figure 4 and recursively call the recursive subroutine.

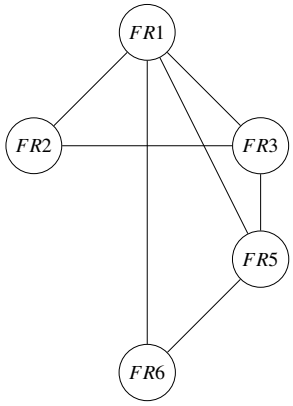


FIGURE 4. FR1 AND FR4 CONTRACTED

24. (sub-1) The new G now has $n = 5$ and $m = 7$. Since $n = 5 > 3 = k$, do not return.
25. (sub-1) Perform the edge density calculation for $n = 5, k = 3, m = 7$:

$$\frac{n}{2k}(kn - n) = \frac{5}{2 \cdot 3}(3 \cdot 5 - 5) = 8.3 \geq 7 = m \qquad (18)$$

Thus, G may be 3-colorable so do not return.
26. (sub-1) Since $\mathrm{d}(FR2) = \mathrm{d}(FR6) = 2 < 3 = k$, replace G with $G - \{FR2, FR6\}$ as show in Figure 5.
27. (sub-1) The new G now has $n = 3$ and $m = 3$. Since $n = 3 \leq 3 = k$, conclude that G is 3-colorable and return true.
28. (sub-0) Return true.
29. (main) Return $\chi(G) = 3$.

For the sake of demonstration, assume that the recursive call after the contraction returns false.

29. (sub-0) Add an edge between $FR1$ and $FR4$, as shown in Figure 6 and recursively call the recursive subroutine.
30. (sub-1) The new G now has $n = 6$ and $m = 10$. Since $n = 6 > 3 = k$, do not return.

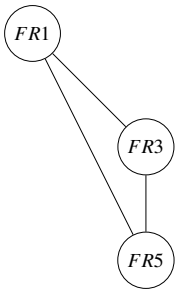


FIGURE 5. FR2 AND FR6 REMOVED

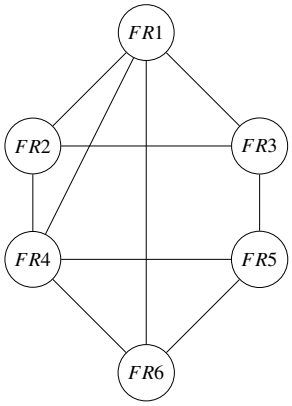


FIGURE 6. FR1-FR4 EDGE ADDED

31. (sub-1) Perform the edge density calculation for $n = 6, k = 3, m = 10$:

$$\frac{n}{2k}(kn-n) = \frac{6}{2\cdot 3}(3\cdot 6-6) = 12 \geq 10 = m \qquad (19)$$

Thus, G may be 3-colorable so do not return.
32. (sub-1) There are no nodes with degree < 3 so continue.
33. (sub-1) Since $N(FR5) \subseteq N(FR1$, replace G with $G-\{FR5\}$ as shown in Figure 7.
34. (sub-1) The new G now has $n = 5$ and $m = 7$. Since $n = 5 > 3 = k$, do not return.
35. (sub-1) Perform the edge density calculation for $n = 5, k = 3, m = 7$:

$$\frac{n}{2k}(kn-n) = \frac{5}{2\cdot 3}(3\cdot 5-5) = 8.3 \geq 7 = m \qquad (20)$$

Thus, G may be 3-colorable so do not return.
36. (sub-1) Since $\mathrm{d}(FR3) = \mathrm{d}(FR6) = 2 < 3 = k$, replace G with $G-\{FR3, FR6\}$ as show in Figure 8.
37. (sub-1) The new G now has $n = 3$ and $m = 3$. Since $n = 3 \leq 3 = k$, conclude that G is 3-colorable and return true.

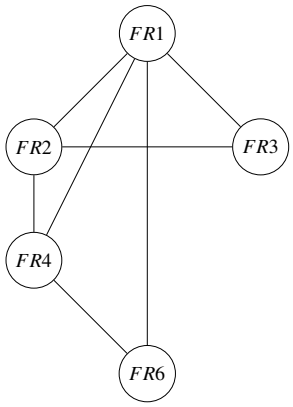


FIGURE 7. FR5 REMOVED

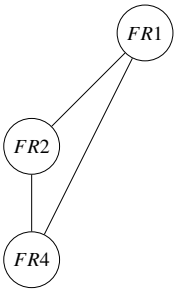


FIGURE 8. FR3 AND FR6 REMOVED

38. (sub-0) Return true.
39. (main) Return $\chi(G) = 3$.

The result of running the algorithm on the sample graph shows that the sample graph is 3-colorable. An example 3-coloring of the sample graph is shown in Figure 9.

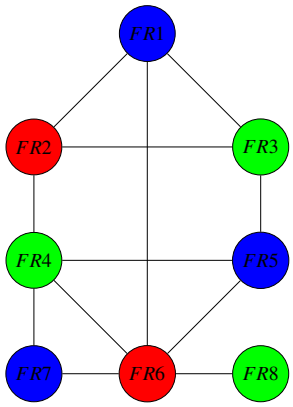


FIGURE 9. SAMPLE 3-COLORING

This means that nodes (FRs) that are assigned the same color can be combined together into a single physical part.

THEORETICAL BASIS

The lemmas and corollaries that support the steps in the algorithm are now presented. All of these proofs actually apply to steps in the recursive subroutine; the main loop is more straightforward.

Edge Density Test (Step 2)

We start with the justification for the edge density threshold.

Lemma 1. *Let G be a graph of order n and size m:*

$$G \text{ is } k\text{-colorable} \implies m \leq \frac{n}{2k}(kn-n).$$

Proof. Assume G is k-colorable. This means that $V(G)$ can be distributed into k independent (some possibly empty) subsets. Call these subsets $A_1, \ldots A_k$ and let $a_i = |A_i|$. Thus, each $v \in A_i$ can be adjacent to at most $n - a_i$ other nodes in G, and hence the maximum number of edges incident to nodes in A_i is given by: $a_i(n-a_i) = na_i - a_i^2$. Now, using the fundamental theorem of graph theory, the maximum number of edges in G is given by:

$$m \leq \frac{1}{2}\sum_{i=1}^{k}(na_i - a_i^2)$$

with the constraint:

$$\sum_{i=1}^{k} a_i = n$$

This problem can be solved using the Lagrange multiplier technique:

$$\frac{1}{2}(n-2a_i) = \lambda$$
$$a_i = \frac{n}{2} - \lambda$$

$$\sum_{i=1}^{k} a_i = \sum_{i=1}^{k}\left(\frac{n}{2}-\lambda\right) = k\left(\frac{n}{2}-\lambda\right) = n$$
$$\lambda = \frac{n}{2} - \frac{n}{k}$$

$$a_i = \frac{n}{2} - \left(\frac{n}{2} - \frac{n}{k}\right) = \frac{n}{k}$$

Therefore:

$$m \leq \frac{1}{2}\sum_{i=1}^{k}\left[n\left(\frac{n}{k}\right) - \left(\frac{n}{k}\right)^2\right] = \frac{k}{2}\left(\frac{n^2k-n^2}{k^2}\right) = \frac{n}{2k}(kn-n)$$

The algorithm actually uses the contrapositive:

Corollary 1. *Let G be a graph of order n and size m and let $k \in \mathbb{N}$:*

$$m > \frac{n}{2k}(kn-n) \implies G \text{ is not } k\text{-colorable.}$$

Low Degree Node Removal (Step 3)

Next is the lemma that justifies removal of low-degree nodes. This lemma requires a small utility lemma, which is presented first:

Lemma 2. *Let G be a graph and let $v \in V(G)$:*

$$G \text{ is } k\text{-colorable} \implies G - v \text{ is } k\text{-colorable.}$$

Proof. Assume G is k-colorable.

Case 1: v has its own unique color.
$G - v$ still has a proper coloring using $k-1$ colors and hence is $(k-1)$-colorable, and thus is k-colorable (Proposition 1).

Case 2: v shares a color with some other node.
$G - v$ still has a proper coloring using k colors and is thus still k-colorable.

$\therefore G - v$ is k-colorable.

Now, we present the actual lemma:

Lemma 3. *Let G be a graph and let $v \in V(G)$ such that $d(v) < k$ for some $k \in \mathbb{N}$:*

$$G \text{ is } k\text{-colorable} \iff G - v \text{ is } k\text{-colorable.}$$

Proof.

$\implies$ Assume G is k-colorable.
$\therefore G - v$ is k-colorable (Lemma 2).

$\impliedby$ Assume $G - v$ is k-colorable. By assumption, $d(v) < k$, meaning v has at most $k-1$ neighbors, using at most $k-1$ colors. Thus, there is always an additional color available for v. So extend $G - v$ to G and color v with one of the available $k - d(v)$ colors. The result is a proper $((k-1)+1 = k)$-coloring of G.
$\therefore G$ is k-colorable.

The algorithm actually uses an inductive corollary that enables all such low-degree nodes to be removed at once:

Corollary 2. *Let G be a graph of order n and let $X = \{v \in V(G) | \, \mathrm{d}(v) < k\}$ for some $k \in \mathbb{N}$:*

G is k-colorable $\iff$ *G − X is k-colorable.*

Proof. (by induction on $|X|$)

1. (Base Case) Let $|X| = 0$.
   Since $G - X = G$, G is k-colorable $\iff$ $G - X = G$ is k-colorable.
2. (Inductive Assumption) Let $|X| = r$.
   Assume G is k-colorable $\iff$ $G - X$ is k-colorable.
3. (Inductive Step) Consider $|X| = r + 1$.
   Since $|X| = r + 1 > 0$, there exists $v \in X$ such that $\mathrm{d}(v) < k$.
   Let $Y = X - \{v\}$ and note that $|Y| = |X| - 1 = (r+1) - 1 = r$.
   So, G is k-colorable $\iff$ $G - v$ is k-colorable (Lemma 3) $\iff$ $(G - v) - Y = G - X$ is k-colorable (inductive assumption).

Therefore, by the principle of induction, G is k-colorable $\iff$ $G - X$ is k-colorable.

Neighborhood Subset Check (Step 4)

Next is the lemma that justifies removal of nodes whose neighborhoods are subsets of other nodes.

Lemma 4. *Let G be a graph and let $u, v \in V(G)$ such that $N(u) \subseteq N(v)$:*

G is k-colorable $\iff$ *G − u is k-colorable*

Proof.

$\implies$ Assume G is k-colorable.
$\therefore G - u$ is k-colorable (Lemma 2).

$\impliedby$ Assume $G - u$ is k-colorable.
Since $N(u) \subseteq N(v)$ and (by definition) $u \notin N(u)$, it must be the case that $u \notin N(v)$ and hence $uv \notin E(G)$. Thus u and v are allowed to have the same color. Furthermore, since every node adjacent to u is also adjacent to v, none of these nodes can have the same color as v. So extend $G - u$ to G and color u with the same color as v. The result is a proper coloring of G using the same k colors.
$\therefore G$ is k-colorable.

Smallest Neighborhood Intersection (Step 5)

Next is the smallest neighbor intersection test. This lemma requires a small utility lemma, which is presented first:

Lemma 5. *Let G be a graph and let $S \subseteq V(G)$ such that $S \neq \emptyset$ and $\forall u, v \in S, uv \notin E(G)$:*

$$(\exists v \in S, \forall w \in V(G) - S, vw \in E(G)) \implies \forall u \in S, N(u) \subseteq N(v)$$

Proof. Assume $\exists v \in S, \forall w \in V(G) - S, vw \in E(G)$. Now, assume $u \in S$:

Case 1: $N(u) = \emptyset$.
Therefore, by definition, $N(u) = \emptyset \subseteq N(v)$.

Case 2: $N(u) \neq \emptyset$.
Assume $w \in N(u)$. This means that $uw \in E(G)$ and hence $w \notin S$. So $w \in V(G) - S$ and thus, by assumption, $vw \in E(G)$. Hence $w \in N(v)$ and therefore $N(u) \subseteq N(v)$.

$\therefore \forall u \in S, N(u) \subseteq N(v)$

Now, we present the actual lemma:

Lemma 6. *Let G be a graph of order n and size m such that $\forall u, v \in V(G), N(u) \not\subseteq N(v)$ and let $k \in \mathbb{N}$ such that $2 \leq k < n$:*

G is k-colorable $\implies$ *$\exists w, z \in V(G)$ such that $|N(w) \cap N(z)| \leq n - 2 - \frac{n-2}{k-1}$.*

Proof. Assume G is k-colorable. This means that $V(G)$ can be distributed into k independent (some possibly empty) subsets $A_1, \ldots, A_k$ such that $a_i = |A_i|$ and $a_1 \geq a_2 \geq \cdots \geq a_k$. Since $n > k$, by the pigeonhole principle, it must be the case that $a_1 \geq 2$. Assume $v \in A_1$.

First, assume by way of contradiction (ABC) that v is adjacent to all other nodes in $V(G) - A_1$. Since $a_1 \geq 2$, there exists $u \in A_1$ such that $u \neq v$ and u is not adjacent to v. Thus, $N(u) \subseteq N(v)$ (Lemma 5), which contradicts the assumption. Therefore, $\exists v' \in V(G) - A_1$ such that $vv' \notin E(G)$. Assume $v' \in A_i$ for some i such that $1 < i \leq k$:

Case 1: $a_i = 1$
By the pigeonhole principle:

$$a_1 \geq \left\lceil \frac{n-1}{k-1} \right\rceil \geq \frac{n-1}{k-1}$$

Now, assume by way of contradiction (ABC) that v' is adjacent to all nodes in $V(G) - A_1 - A_i$ and assume $u \in N(v)$. Then it must be the case that $u \in V(G) - A_1 - A_i$ and so $uv' \in E(G)$ and thus $u \in N(v')$. Therefore $N(v) \subseteq N(v')$, which contradicts the assumption. This means that there exists some $u \in V(G) - A_1 - A_i$ such that $uv' \notin E(G)$. This results in the upper bound:

$$|N(v) \cap N(v')| \leq n - 2 - \frac{n-1}{k-1}$$

Comparing this to the desired bound:

$$\left(n - 2 - \frac{n-2}{k-1}\right) - \left(n - 2 - \frac{n-1}{k-1}\right) = \frac{1}{k-1} > 0$$

for $k \geq 2$. Thus the new bound is tighter and so:

$$|N(v) \cap N(v')| \leq n - 2 - \frac{n-1}{k-1} \leq n - 2 - \frac{n-2}{k-1}$$

Case 2: $a_i = 2$

By the pigeonhole principle:

$$a_1 \geq \left\lceil \frac{n-2}{k-1} \right\rceil \geq \frac{n-2}{k-1}$$

This results in the upper bound:

$$|N(v) \cap N(v')| \leq n-2-\frac{n-2}{k-1}$$

Case 3: $a_i \geq 3$

By the pigeonhole principle:

$$a_1 \geq \left\lceil \frac{n-3}{k-1} \right\rceil \geq \frac{n-3}{k-1}$$

This results in the upper bound:

$$|N(v) \cap N(v')| \leq n-3-\frac{n-3}{k-1}$$

Comparing this to the desired bound:

$$\left(n-2-\frac{n-2}{k-1}\right)-\left(n-3-\frac{n-3}{k-1}\right) = 1-\frac{1}{k-1} \geq 0$$

for $k \geq 2$. Thus the new bound is tighter and so:

$$|N(v) \cap N(v')| \leq n-3-\frac{n-3}{k-1} \leq n-2-\frac{n-2}{k-1}$$

$\therefore \exists w,z \in V(G)$ such that $|N(w) \cap N(z)| \leq n-2-\frac{n-2}{k-1}$.

The algorithm actually uses the contrapositive:

Corollary 3. *Let G be a graph of order n and size m such that $\forall u,v \in V(G), N(u) \not\subseteq N(v)$ and let $k \in \mathbb{N}$ such that $2 \leq k < n$:*

$$\left(\forall w,z \in V(G), |N(w) \cap N(z)| > n-2-\frac{n-2}{k-1}\right) \implies$$

G is not k-colorable.

Recursive Step (Step 6)

Finally, we present the lemma for the recursive call.

Lemma 7. *Let G be a graph of order $n >= 2$ and let $u,v \in G$ such that $uv \notin E(G)$:*

$$G \text{ is } k\text{-colorable} \iff G \cdot uv \text{ or } G+uv \text{ is } k\text{-colorable.}$$

Proof.

$\implies$ Assume G is k-colorable.

Case a: u and v have the same color.

Then $\forall w \in N(u) \cup N(v)$ it must be the case that w is a different color than the color of u and v. Let v' be the contracted node, so that $N(v') = N(u) \cup N(v)$, and color v' with the same color as u and v. The result is a proper k-coloring of $G \cdot uv$.

$\therefore G \cdot uv$ is k-colorable.

Case b: u and v have the different colors in c.

By adding edge uv, u and v become adjacent and thus must have different colors. Therefore, u and v can retain their same colors. The result is a proper k-coloring of $G+uv$.

$\therefore G \cdot uv$ or $G+uv$ is k-colorable.

$\impliedby$ Assume $G \cdot uv$ or $G+uv$ is k-colorable.

Case a: $G \cdot uv$ is k-colorable.

Let v' be the contracted node with some assigned color. It must be the case that $\forall w \in N(v'), w$ has a different color than v'. Expand $G \cdot uv$ to G and color u and v with the same color as v'. The result is a proper k-coloring of G.

Case b: $G+uv$ is k-colorable.

Remove edge uv. Since u and v are no longer adjacent, there are no requirements on their colors. Thus, they can retain their original colors. The result is a proper k-coloring of G.

$\therefore G$ is k-colorable.

ALGORITHM COMPLEXITY

The running time of the proposed algorithm is determined by Theorem 3.

Theorem 3. *Let G be a graph of order n and size m. Using the above algorithm, we can determine in time $2^{\frac{n}{2k}(kn-n)-m}$ if G is k-colorable or not.*

Proof. We apply induction on $\frac{n}{2k}(kn-n)-m$. If $m > \frac{n}{2k}(kn-n)$ then G is not k-colorable by Lemma 1. Thus, assume that $m \leq \frac{n}{2k}(kn-n)$.

To find the degrees of the nodes, the complexity would be at most n^2. If G has a node v of degree at most $k-1$ then by the induction hypothesis we can determine in time $2^{\frac{n-1}{2k}(k(n-1)-(n-1)-k)-m+k-1}$ if $G-v$ is k-colorable. As a result, we can determine in time $n^2 + 2^{\frac{n-1}{2k}(k(n-1)-(n-1))-m+k-1} < 2^{\frac{n}{2k}(kn-n)-m}$ if G is k-colorable, as desired.

To check if G has a pair of nodes u and v such that $N(u) \subseteq N(v)$, we need at most n^3 operations. If G has nodes u and

v such that $N(u) \subseteq N(v)$, then by the induction hypothesis we can determine in time $2^{\frac{n-1}{2k}(k(n-1)-(n-1))-m+k-1}$ if $G-u$ is k-colorable. As a result, by Lemma 4, we can determine in time $n^3+2^{\frac{n-1}{2k}(k(n-1)-(n-1))-m+k-1} < 2^{\frac{n}{2k}(kn-n)-m}$ if G is k-colorable, as desired.

Now suppose u and v are are nodes in G with the smallest number of common neighbors in G. The cost to find this pair of nodes in G is at most n^3. By Lemma 6 we have $|N(u) \cap N(v)| \leq n-2-\frac{n-2}{k-1}$.

By Lemma 7, G is k-colorable iff $G \cdot uv$ or $G+uv$ are k-colorable. Note that $G \cdot uv$ has $n-1$ nodes and at least $m-n+2+\frac{n-2}{k-1}$ edges. Moreover, $G+uv$ has n nodes and $m+1$ edges. Therefore, we can apply the induction hypothesis on the graphs $G \cdot uv$ and $G+uv$. As a result, by the induction hypothesis, we can determine in time:

$$n^3+2^{\frac{n-1}{2k}(k(n-1)-(n-1))-m+n-2-\frac{n-1}{k-1}}+2^{\frac{n}{2k}(kn-n)-m+1} \quad (21)$$

$$= n^3+2^{\frac{n}{2k}(kn-n)-m-1}\left(2^{\frac{n}{k}-\frac{1}{2k}-\frac{n-2}{k-1}}+1\right) \quad (22)$$

$$< 2^{\frac{n}{2k}(kn-n)-m} \quad (23)$$

if G is k-colorable or not, as desired.

This expression predicts that the sample graph with $n=9$ and $k=3$ will take at most 1024 steps.

Looking at the edge cases, a complete graph should be on the order of n steps, since the edge density test will fail on each iteration until $k=n$. The expression predicts on the order of 1 step. An empty graph should be on the order of n^2 steps, since low degree removal will remove all the nodes on the first iteration. The expression predicts on the order of 1 step.

The differences can be attributed to the fact that no recursion is required for these edge cases, and so the exponential terms in the proof are dominated by the polynomial time steps. Thus, the derived expression best fits when recursion is necessary.

EXAMPLE: TOASTER DESIGN

In this section, we describe the application of the proposed algorithm to the example of a toaster. The objective is to reduce the number of parts. Initially, we consider a toaster with 9 FRs such that each FR is assigned to a separate part — thus 9 parts.

Figure 10 shows two examples of toasters available in the market. The goal is to reduce the number of parts from 9 to some value less than 9 by using the proposed algorithm to partition the FRs into the lower number of parts.

The following functional requirements are defined for the toaster:

FR1: Advancing of bread (press down)
FR2: Toaster ON

FIGURE 10. TOASTER EXAMPLES

FR3: Toaster start
FR4: Grip for comfort
FR5: Toaster stop
FR6: Temperature control
FR7: Bread holder
FR8: Body must accommodate all parts
FR9: Bush for surface to prevent scratches

We consider three different scenarios (i.e., connections between FRs) for the same number of FRs. The designer needs to connect the FRs together in the initial graph based on the desire to integrate such FRs together. Table 1 illustrates the steps of the algorithm for the first case. Table 2 shows the resulting graphs for the other two cases. Finally, Tables 3, 4, and 5 show how the FRs are arranged together based on the results obtained.

Figure 11 shows a general procedure that can be followed to determine the minimum number of parts given a set of FRs and the connection between them.

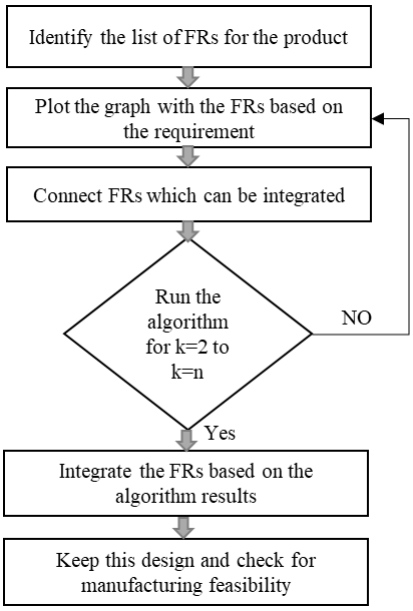
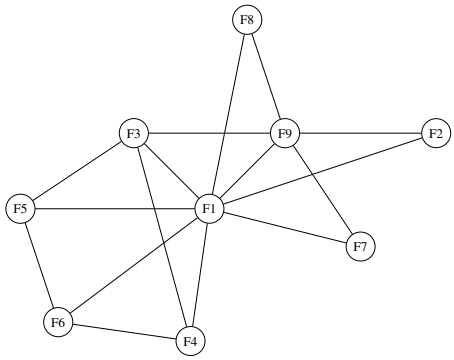
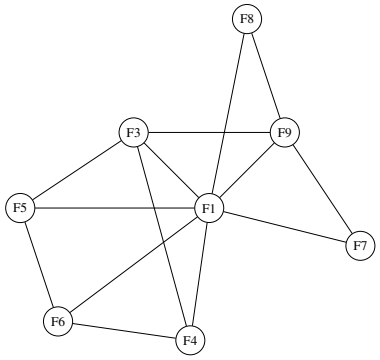
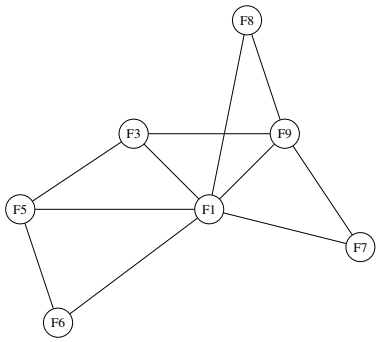


FIGURE 11. OVERALL PROCEDURE

TABLE 1: TOASTER EXAMPLE CASE 1

$k = 1$ $n = 9$ $m = 16$	
(1): $n = 9 > 1 = k$, so continue. (2): $\frac{9}{2 \cdot 1}(1 \cdot 9 - 9) = 0 < 16 = m$, so set $k = 2$. (1): $n = 9 > 2 = k$, so continue. (2): $\frac{9}{2 \cdot 2}(2 \cdot 9 - 9) = 20.25 \geq 16 = m$, so continue. (3): The degrees of all the nodes are $>= 2$, so continue. (4): $N(F2) \subseteq N(F3)$, so replace G with $G - \{F2\}$.	
(1): $n = 8 > 2 = k$, so continue. (2): $\frac{8}{2 \cdot 2}(2 \cdot 9 - 9) = 18 \geq 14 = m$, so continue. (3): The degrees of all the nodes are $>= 2$, so continue. (4): $N(F4) \subseteq N(F5)$, so replace G with $G - \{F4\}$.	

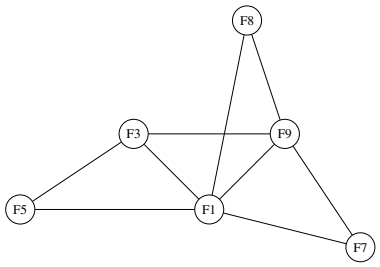
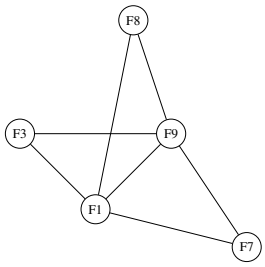
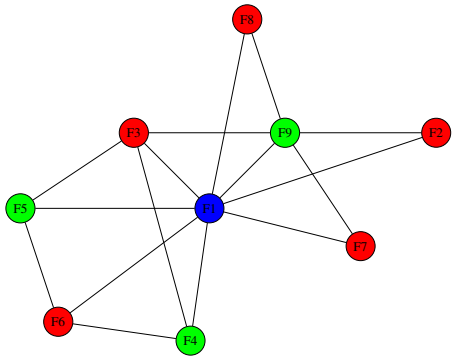
(1): $n = 7 > 2 = k$, so continue. (2): $\frac{7}{2 \cdot 2}(2 \cdot 7 - 7) = 24.5 \geq 11 = m$, so continue. (3): The degrees of all the nodes are >= 2, so continue. (4): $N(F6) \subseteq N(F3)$, so replace G with $G - \{F6\}$.	
(1): $n = 6 > 2 = k$, so continue. (2): $\frac{6}{2 \cdot 2}(2 \cdot 6 - 6) = 9 \geq 9 = m$, so continue. (3): The degrees of all the nodes are >= 2, so continue. (4): $N(F5) \subseteq N(F9)$, so replace G with $G - \{F5\}$.	
(1): $n = 5 > 2 = k$, so continue. (2): $\frac{5}{2 \cdot 2}(2 \cdot 5 - 5) = 6.25 < 7 = m$, so set $k = 3$. (1): $n = 5 > 3 = k$, so continue. (2): $\frac{5}{2 \cdot 3}(3 \cdot 5 - 5) = 8.33 \geq 7 = m$, so continue. (3): The degrees of F1,F7,F8 $< 3 = k$ so replace G with $G - \{F1, F7, F9\}$.	
(1): $n = 1 \leq 3 = k$, so return true (G is 3-colorable). An example 3-coloring is shown.	

TABLE 2. TOASTER EXAMPLE CASES 2 AND 3

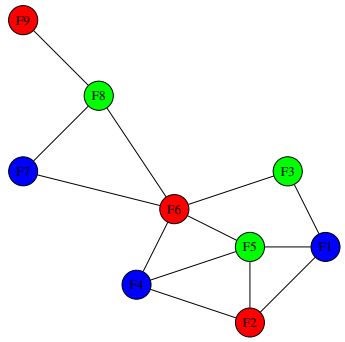
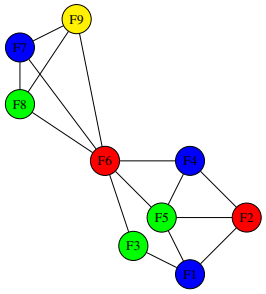
Case 2: 3-colorable ($n = 9$ and $m = 13$)	
Case 3: 4-colorable ($n = 9$ and $m = 15$)	

TABLE 3. ASSIGNING FRs TO PARTS FOR CASE 1

Case 1 **3-colorable**
Part 1: FR1:Advancing of bread (press down) **Part 2:** FR2:Toaster ON FR3:Toaster start FR6:Temperature control FR7:Bread holder FR8:Body must accommodate all parts **Part 3:** FR4:Grip for comfort FR5:Toaster Stop FR9:Bush for surface to prevent scratches

TABLE 4. ASSIGNING FRs TO PARTS FOR CASE 2

Case 2 **3-colorable**
Part 1: FR1:Advancing of bread (press down) FR4:Grip for comfort FR7:Bread holder **Part 2:** FR2:Toaster ON FR6:Temperature control FR9:Bush for surface to prevent scratches **Part 3:** FR3:Toaster start FR5:Toaster Stop FR8:Body must accommodate all parts

TABLE 5. ASSIGNING FRs TO PARTS FOR CASE 3

Case 3 **4-colorable**
Part 1: FR1:Advancing of bread (press down) FR4:Grip for comfort FR7:Bread holder **Part 2:** FR2:Toaster ON FR6:Temperature control **Part 3:** FR3:Toaster start FR5:Toaster Stop FR8:Body must accommodate all parts **Part 4:** FR9:Bush for surface to prevent scratches

CONCLUSION AND FUTURE WORK

The objective of the study is to develop a graph coloring technique to identify the minimum number of parts needed to satisfy a list of FRs and the connections between them. The proposed algorithm was applied to a toaster design with 9 FRs for 3 different configurations. The results showed that a 2-part design is not possible to cover 9 FRs, but a 3-part design is achievable. The proposed method has potential applications, particularly in the additive manufacturing industry, which has the capability of printing products with desired shape and complexity.

This work can be extended in several ways. First, the algorithm can be employed for more complex systems and the efficiency can be compared with different types of algorithms. Second, the algorithm can be extended to consider the connection between FRs based on design parameters, process variables, and process control factors. Finally, the algorithm can be used in different manufacturing contexts such as additive manufacturing and smart manufacturing and its capabilities can be tested at the process level as well as the design level.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation–USA under grants #CMMI-1727190 and CMMI-1727743. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Shirwaiker, R. A., and Okudan, G. E., 2008. "Triz and axiomatic design: a review of case-studies and a proposed synergistic use". *J. Intell. Manuf.,* **19**(1), Feb, pp. 33–47.

[2] P., Y. V., and Zakrevskij, A. A. S. A. D., 1986. "Applications of graph theory to the problems of logical design". *Investig. Appl. Graph Theory*, pp. 3–9.

[3] Yang, S., Santoro, F., and Zhao, Y. F., 2018. "Towards a numerical approach of finding candidates for additive manufacturing-enabled part consolidation". *J. Mech. Des.,* **140**(4), p. 41701.

[4] Schmelzle, J., Kline, E. V., Dickman, C. J., Reutzel, E. W., Jones, G., and Simpson, T. W., 2015. "(re) designing for part consolidation: understanding the challenges of metal additive manufacturing". *J. Mech. Des.,* **137**(11), p. 111404.

[5] Yang, S., Tang, Y., and Zhao, Y. F., 2015. "A new part consolidation method to embrace the design freedom of additive manufacturing". *J. Manuf. Process.,* **20**, pp. 444–449.

[6] Tang, Y., Yang, S., and Zhao, Y. F., 2016. "Sustainable design for additive manufacturing through functionality integration and part consolidation bt". In *Handbook of Sustainability in Additive Manufacturing*, S. S. Muthu and M. M. Savalani, eds., Vol. 1. Springer, Singapore, pp. 101–144.

[7] Rong, Z., Yang, Z., Li, Y., Chen, K., and Dan, B., 2017. "Modular product design based on the supply chain network". *Adv. Mech. Eng.,* **9**(10), Oct, p. 1687814017732308.

[8] Simpson, T. W., Jiao, J., Siddique, Z., and Hölttä-Otto, K., 2014. *Advances in product family and product platform design*. Springer, New York.

[9] Reichmann, D., Rahat, O., Albeck, S., Meged, R., Dym, O., and Schreiber, G., 2005. "The modular architecture of protein–protein binding interfaces". *Proc. Natl. Acad. Sci. U. S. A.,* **102**(1), Jan, pp. 57–62.

[10] Kaksonen, M., Toret, C. P., and Drubin, D. G., 2005. "A modular design for the clathrin-and actin-mediated endocytosis machinery". *Cell,* **123**(2), pp. 305–320.

[11] Lunde, B. M., Moore, C., and Varani, G., 2007. "Rna-binding proteins: modular design for efficient function". *Nat. Rev. Mol. cell Biol.,* **8**(6), p. 479.

[12] Dahmus, J. B., Gonzalez-Zugasti, J. P., and Otto, K. N., 2001. "Modular product architecture". *Des. Stud.,* **22**(5), pp. 409–424.

[13] Tseng, H. E., Chang, C. C., and Li, J. D., 2008. "Modular design to support green life-cycle engineering". *Expert Syst. Appl.,* **34**(4), pp. 2524–2537.

[14] Gu, P., Hashemian, M., Sosale, S., and Rivin, E., 1997. "An integrated modular design methodology for life-cycle engineering". *CIRP Ann.,* **46**(1), pp. 71–74.

[15] Zetterberg, A., Mörtberg, U. M., and Balfors, B., 2010. "Making graph theory operational for landscape ecological assessments, planning, and design". *Landsc. Urban Plan.,* **95**(4), pp. 181–191.

[16] Papalambros, P. Y., 1995. "Optimal design of mechanical engineering systems". *J. Vib. Acoust.,* **117**(B), pp. 55–62.

[17] Bondy, J. A., and Murty, U. S. R., 1976. "Graph theory with applications". *Citeseer,* **290**.

[18] Galinier, P., and Hertz, A., 2006. "A survey of local search methods for graph coloring". *Comput. Oper. Res.,* **33**(9), pp. 2547–2562.

[19] Descartes, B., 1977. "Ja bondy and usr murty, graph theory with applications". *Bull. Am. Math. Soc.,* **83**(3), pp. 313–315.

[20] Li, M., Zhang, Y. F., and Fuh, J. Y. H., 2010. "Retrieving reusable 3d cad models using knowledge-driven dependency graph partitioning". *Comput. Aided. Des. Appl.,* **7**(3), pp. 417–430.

[21] Wolfe, G., Wong, J. L., and Potkonjak, M., 2002. "Watermarking graph partitioning solutions". *IEEE Trans. Comput. Des. Integr. Circuits Syst.,* **21**(10), pp. 1196–1204.

[22] Wakabayashi, S., Isomoto, K., Koide, T., and Yoshida, N., 1994. "A systolic graph partitioning algorithm for vlsi design". Vol. 1 of *Proceedings of the IEEE International Symposium on Circuits and Systems ISCAS'94*, pp. 225–228.

[23] Li, F., Zhang, Q., and Zhang, W., 2007. "Graph partitioning strategy for the topology design of industrial network". *IET Commun.,* **1**(6), pp. 1104–1110.

[24] Kim, N., Zheng, Z., Elmetwaly, S., and Schlick, T., 2014. "RNA graph partitioning for the discovery of RNA modularity: A novel application of graph partition algorithm to biology". *PLoS One,* **9**(9).

[25] Zhang, C., Zhang, Y., Li, D., Li, J., and Li, M., 2016. "Smartpartition: Efficient partitioning for natural graphs". Proceedings of the IEEE International Conference on Cluster Computing 2016, pp. 130–131.

[26] Jiang, G., Wu, J., Lam, S. K., Srikanthan, T., and Sun, J., 2015. "Algorithmic aspects of graph reduction for hardware/software partitioning". *J. Supercomput.,* **71**(6), pp. 2251–2274.

[27] Dongen, S. M. V., 2000. "Graph clustering by flow simulation". PhD Thesis, University of Utrecht, May. See also URL `https://dspace.library.uu.nl/handle/1874/848`.

[28] Brandes, U., Gaertler, M., and Wagner, D., 2003. "Experiments on graph clustering algorithms". Proceedings of the European Symposium on Algorithms, pp. 568–579.

[29] Assayony, M. O. H., 2008. "Design of a parallel graph-based protein sequence clustering algorithm". Vol. 3 of *Proceedings of the International Symposium on Information Technology ITSim 2008*, pp. 1–8.

[30] Suh, N. P., 1998. "Axiomatic design theory for systems". *Research in Engineering Design,* **10**, pp. 189–209.

[31] Boothroyd, G., Dewhurst, P., and Knight, W., 2010. *Prod-*

uct design for manufacture and assembly. CRC Press, Boca Raton.

[32] Gopalakrishnan, P. K., Kein, H., Jahanbekam, S., and Behdad, S., 2018. "Graph partitioning technique to identify physically integrated design concepts". Proceedings of the ASME 2018 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE 2018.