IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 1, 2021, accepted January 26, 2022, date of publication February 4, 2022, date of current version February 17, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3149001

Complex Network–Based Change Propagation Path Optimization in Mechanical Product Development

YONG YIN[1,2], SHUXIN WANG[3], AND JIAN ZHOU[1]

[1]Key Laboratory of Fiber Optic Sensing Technology and Information Processing, Ministry of Education, Wuhan University of Technology, Wuhan 430070, China
[2]Shenzhen Research Institute, Wuhan University of Technology, Shenzhen 518000, China
[3]Wenzhou University of Technology, Wenzhou 325027, China

Corresponding author: Shuxin Wang (wangshuxin@wzu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 51875429, in part by the General Program of Shenzhen Natural Science Foundation under Grant JCYJ20190809142805521, and in part by the Research Project of Wuhan University of Technology Chongqing Research Institute under Grant YF2021-18.

ABSTRACT Change propagation is a process in which a change to one part or element of a developed mechanical product tends to induce additional changes to other elements of the product, and the changes can propagate further iteratively. Different change propagation paths yield different change fulfillment costs (development expenditures, lead times, quality losses, etc.). In this study, we seek an optimal change propagation path based on complex theories, and by using this approach the cost of the change can be minimized. In particular, we introduce a modified Dijkstra algorithm coupled with a complex network described based on two variables: change probability and change impact. The network depicting the components of a mechanical product and their dependencies is directional while considering both node weights and edge weights. The roles of the components such as change absorbers, carriers, and multipliers are identified. Moreover, the loop change propagation paths and "AND" logical relations between sibling components are investigated. The practical application of the proposed models and methods is assessed using an industrial case example of an elevator system, and satisfactory results are obtained. The proposed method can be employed for analyzing different change propagation paths and their optimization in terms of overall costs.

INDEX TERMS Change propagation path, complex network, dependencies, modified Dijkstra algorithm.

I. INTRODUCTION

Manufacturing enterprises have faced unprecedented challenges owing to increased consumption diversity and market uncertainty. Thus, changes in product development processes are inevitable. They are usually classified into two major categories: initiated development changes and emergent changes [1]. The former is often driven by new customer demands (product size, style, weight, etc.), new environmental constraints (temperature, humidity, vibration, etc.), technological innovations, and certification requirements at the early development stages. The latter often originates from mistakes that may not be revealed until late product development stages and changes are ineluctably made to fix the mistakes [2].

The associate editor coordinating the review of this manuscript and approving it for publication was Chao-Yang Chen.

Generally, a complex product contains thousands of parts or components that are connected with each other in different ways. In this study, we refer to parts and components as components for simplicity. Components are the smallest units that cannot be subdivided further. Thus, changes in one component are likely to induce changes in other components and they can propagate further iteratively. Different change propagation paths afford different fulfillment costs (development expenditures, lead times, quality losses, etc.). Thus, the optimization of change propagation paths has attracted considerable research attention.

Change propagation refers to a process in which a change in one component of a product may induce additional changes in its downstream components, even though the triggered changes are unexpected. Change propagation often proceeds in a cause–effect–cause–effect manner until the process reaches a stable state [3]. The situation becomes even

more complex when a change propagation occurs during the development of highly complex products [4]. Here, the complexity denotes the structural complexity of the components and complex connections between them. Studies indicate that 32.4% of the total change in the components of a complex product is attributed to propagated changes [5], [6]. The optimization of change propagation paths can provide considerable benefits, including decreased lead time and expenditure cost and improved quality of the product development process. Therefore, it is important to find the optimal change propagation paths [7].

In this study, we identify an optimal change propagation path triggered by an initial change. Using this approach, the cost of the change can be minimized. The remainder of this paper is organized as follows. In Section 2, we summarize recent advancements and key challenges from the change propagation path optimization. A detailed analysis of change propagation is presented in Section 3. Section 4 introduces the network construction process. In Section 5, the change propagation path optimization algorithm, namely, the modified Dijkstra algorithm is introduced. Subsequently, Section 6 demonstrates a case study of an elevator system design to verify the proposed models and methods. Finally, Section 7 summarizes the results of this study and provides future outlook.

II. RELATED RESEARCH

The study on changes can be traced back to the early 1980s [8]. Among various methods, the design structure matrix (DSM) has been regarded as a good roadmap to address changes. The DSM has proven to be very valuable in understanding, designing, and optimizing complex system architectures such as those of products, organizations, and processes, and has been employed in building construction, semiconductor production lines, aerospace development and assembling, large-scale product development, etc [9]. However, some inherent limitations still restrict its further applications [10]. The DSM is based on a priori assumption such that the processes in the system are well defined; thus, DSM-based models cannot be expanded or refined. Moreover, in the DSM, dependencies between connected parts/entities/activities are not straightforward, which results in redundant and parallel dependency paths cannot be distinguished. Axiomatic design (AD) is another popular matrix-based method for dependency and change propagation analysis in a 0–1 binary form. AD uses matrix methods for systematically transforming customers' needs into functional requirements, design parameters, and process variables [11], [12]. AD can be used to assess whether a product development or a system design is good or bad based on the mappings between each functional requirement and design parameter of a matrix; however, it cannot guide coupled design tasks, let alone the analysis and measurement of an interaction strength [13], [14].

The project evaluation and review technique (PERT) is a statistical tool for project or process management. It can be used for analyzing and representing tasks involved in completing a project and has attracted research attention [15]. A PERT diagram can be used to visually identify the critical path and potentially reduce the project duration. The latter is achieved based on a better understanding of dependencies and thus has been successfully coupled and used with the critical path method (CPM) [16]. However, the PERT has limitations. For example, it cannot represent a design process with iteration loops because it was initially developed for linear processes [17]. Later, the Graphical Evaluation and Review Technique (GERT), one of the intriguing techniques used for network-based management, has received increased attention. It owns excellent features such as stochastic models, feedback loops and multiple sink nodes which are impossible in PERT/CPM [18]. However, drawing and calculation is much time-consuming in GERT, and it is unable to deal with dependence of multiple random variables [19].Several other approaches have been proposed to address dependency and change propagation analysis. Ollinger and Stahovich [20] introduced the Redesign IT tool for managing design changes. Possible side effects originating from a design change can be determined using this tool. Chungyu *et al.* [21] developed an analytic hierarchical process method for estimating the strength of functional couplings and change propagation with fuzzy control. Li and Chen [22] suggested a design dependency matrix to examine the dependencies between parameter relations and functions. Lee *et al.* [23] proposed an analytic network process (ANP) approach for explaining the dependency strength between parts and modules in a modular product and recommended the use of this approach for controlling design change impacts and change propagation because of part dependencies. However, this approach involves high computations when the number of parts increases.

The aforementioned approaches are roughly based on matrix models. However, certain congenital defaults exist in matrix-based approaches. Hence, network-based approaches for dependency and change propagation issues have been recently attracting increasing attention in the engineering community. The Petri net is considered the preliminary and formal modeling tool for describing distributed or parallel processing systems [24]. It offers a practical graphical method for capturing and analyzing the dynamic states of the system. However, the Petri net is incompatible with complex problems. It tends to become excessively large even for a modest-sized system [25].

More recently, research on complex networks has unified scholars from different disciplines including mathematics, physics, sociology, manufacturing science, and computer science. Further, researchers have considered the theory of complex networks and gradually adopted it to describe and solve dependency and change propagation problems in complex manufacturing systems. For example, Yu *et al.* proposed a network-based method for analyzing change impacts in customized complex product development and developed a change propagation searching model based on the Matthew

effect theory [26]. Cheng and Chu presented three changeability indices (degree changeability, reach changeability, and between changeability) based on the weighted network theory [2]. Lee *et al.* proposed an ANP approach for estimating the relative importance of parts and modules in a modular product in terms of design change impacts and propagation. [23]. Gong *et al.* analyzed the indices to evaluate the importance of nodes and studied a method for classifying the nodes of a product development network [27]. Ma, Jiang, and Liu built a design change analysis model based on the design property network. They defined a change propagation intensity (CPI) by quantifying the change propagation impact and sought the optimal influence propagation path corresponding to the minimized maximum of the accumulated CPI [28]. Qin *et al.* introduced a weighted network for multistage machining processes based on a complex network and proposed a method of variation propagation analysis and variation source identification based on the manufacturing cost. [29].

Based on this literature review, many fruitful results have been achieved when exploring dependency and change propagation problems using complex networks. However, a lot of challenges still arise when applying the existing methods in the optimization of change propagation paths during product development.

(1) When the complex network theory is used to determine the optimal change propagation path, the shortest path is often considered as the optimal one. However, the shortest path may not be the optimal one because the components may serve as change absorbers and absorb more changes than they themselves cause, may serve as change carriers that take the same number of changes as they cause themselves, or may serve as change multipliers and generate more changes in other systems than they require themselves [30]. This case is different from those reported in other studies [2], [3].

(2) Previous studies on the optimization of change propagation paths did not consider loop paths when seeking the shortest change path. They only searched for change propagation paths along upstream or downstream routes (in one direction). However, loop paths do exist, which was not considered in the past, and can be optimal. In other words, existing studies were ineffective in determining the optimal change propagation path by considering loop paths.

(3) When seeking the optimal change propagation path, ''AND'' logical relations between sibling components are not considered. Moreover, the case where at least two sibling nodes of an upstream node must change simultaneously for the change propagating from their common father node has not been considered in the literature to the best of our knowledge.

In this study, we develop a framework to determine the optimal change propagation path during the development of a mechanical product based on the complex network theory. Changes propagating along the optimal paths can fulfill the change process at the minimal cost. The network is directional while considering both node weights and edge weights. Moreover, we investigate loop paths formed by links between components that may constitute propagation paths. Finally, we consider ''AND'' logical relations between sibling components in a change propagation process.

III. MODELS AND ANALYSIS

A. A DEMONSTRATIVE EXAMPLE

Fig. 1 presents a demonstrative application example from the literature [31], illustrating change propagation for a given design scenario. A Sallen–Key low-pass filter product includes a unity-gain amplifier, a signal source (S), two resistors (R_1 and R_2), and two capacitors (C_1 and C_2).

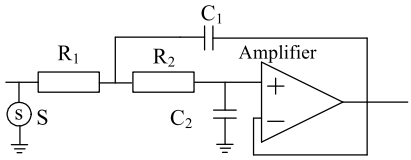


FIGURE 1. A demonstrative application example of the Sallen–Key low-pass filter product.

The performance of the filter is characterized using two parameters, cutoff frequency (ω_c) and quality factor (Q), which are expressed as

$$\begin{cases} \omega_c = \dfrac{1}{2\pi\sqrt{R_1R_2C_1C_2}} & (1) \\ Q = \dfrac{\sqrt{R_1R_2C_1C_2}}{C_2(R_1+R_2)} & (2) \end{cases}$$

Initially, the filter exhibits a baseline ω_c of ω zero kHz and a Q of 0.5. However, a customer demands a change in ω_c to 5 kHz while retaining Q at 0.5. This goal can be achieved using several possible solutions. In other words, two or more alternative schemes involving different components can achieve the same goal. For example, the R_1 value can be doubled provided that R_1 equals R_2, the C_1 value can be doubled by assuming C_1 equals C_2, or both the C_1 and C_2 values are doubled if C_1 and C_2 are unequal, etc.

Fig. 1 demonstrates that R_1, R_2, C_1, and C_2 are the components of the filter and are ''linked'' through electrical parameters. The change requirement (CR) in ω_c from a customer ωiggers a change in one component, which in turn induces additional changes in other components until the circuit reaches a new stability. This example reveals that a CR is often associated with several components. Different choices of which components should change yield different change propagation paths, which correspondingly involve different consumptions of different resources. Therefore, investigating how a change may affect other components and ways to control the change propagation paths with minimal costs are required.

B. RELATION BETWEEN COMPONENTS

A complex product involves thousands of components. These components are linked with each other in various ways such

as in terms of geometric constraints (length, breadth, thickness, depth, etc.), machining sequences (serial or parallel and upstream or downstream), material relations (expansion coefficient, heat transfer coefficient, elastic modulus, allowable stiffness, etc.), and environmental conditions (humidity, temperature, air velocity, etc.). If components of a complex product are treated as nodes and the relation links between each component are regarded as edges, the components and their links constitute a complex network. Thus, the complex network theory can be an effective tool for investigating change propagation paths.

There are two feasible modes for developing a dependency network for two types of linking relations: functional and structural relations. The former are often mapped in the form of parameters, while the latter are associated with component locations in the product. Based on these two types of linking relations, two propagation patterns are possible: parameter propagation pattern and topological face propagation pattern. For the former, changes in parameters propagate to exogenous parameters. Alternatively, for the latter, the changed component propagates topological face changes to other components so as to meet assembly requirements. However, an in-depth analysis reveals that changes in assembly requirements are related to geometric structure parameters such as diameter, stroke, thickness, and length. Thus, all change propagations can be attributed to parameter links between components. This details parameter relation network construction approaches that we use in this study.

C. PARAMETER CHANGE PROPAGATION PATTERNS

Once an initial change in the component of a product is triggered, change propagation becomes inevitable and can induce further changes in other components. Fig. 2 presents the process of change propagation [32]. A CR induces a series of subsequent changes in components $A1$ and $A2$ which are considered to be the components that originate the change propagation. Each originating change component corresponds to different change propagation paths.

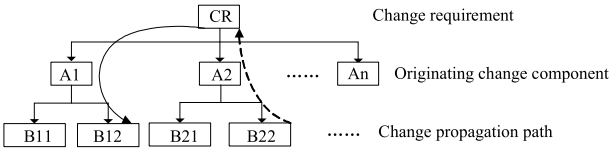


FIGURE 2. Process of change propagation.

The parameter change relation between components can be expressed as $y = f(x_1, x_2, \ldots, x_i, \ldots, x_n)$, where y denotes the changed parent parameter and x_i represents the i^{th} child changed parameter. The parent parameter is dependent, and its value is determined by interdependent child parameters. In other words, a CR from the parent parameter must demand changes in its one or more child nodes. However, as changes in the child nodes can offset one another, the parent parameter may remain the same when changes occur in its child nodes. For example, even if both the C_1 and C_2 values are doubled, Q remains the same (Eq. (2)).

Generally, three categories of nodes can be found: root, transition, and leaf nodes [3]. Root nodes are present at the top of the change network and only act as parents. To facilitate the analysis of parameter change propagation patterns, CRs are generally treated as the root nodes (Fig. 2), similar to ω_c and Q in the Sallen–Key low-pass filter product. Moreover, the root nodes cannot be changed directly and must vary with their child nodes. For example, the ω_c and Q of the Sallen–Key low-pass filter product are typical parent node and are determined using their child nodes R_1, R_2, C_1, and C_2 based on Eqs. (1) and (2). Conversely, leaf nodes are the smallest units of a change. They are present at the bottom of the change path and can only play the role of child nodes, similar to $B11$ and $B12$ in Fig. 2. Transition nodes exist between the root and leaf nodes, functioning as the parent for the downstream nodes or a child for the upstream nodes. For example, $A1$ and $A2$ in Fig. 2 are typical transition nodes.

Changes can propagate in different directions, considering initially changed nodes at different locations. If the change propagates from the root nodes to the child or transition nodes, it is a downstream propagation. For example, the change propagation path from the CR to $B12$ via $A1$ is a downstream propagation, represented by the solid arc with an arrow (Fig. 2). Alternatively, a change propagation from the child or transition nodes to the root nodes is an upstream propagation, indicated by the dotted arc with a directed arrow in Fig. 2.

To determine the optimal change propagation path, ''AND/OR'' logical relations are employed between sibling nodes to analyze the propagation patterns [32]. The ''AND'' logical relation indicates that more than one downstream sibling component will be simultaneously affected by a change in the same upstream parent component. As shown in Fig. 3(a), the change in $A1$ unavoidably triggers simultaneous changes in both $B11$ and $B12$, indicating the existence of ''AND'' logical relation between them. Alternatively, an ''OR'' logical relation implies that one downstream component is sufficient to fulfill the change that propagates from an upstream component. For example, $A1$ induces a change in only one of the sibling components ($B11$ and $B12$); thus, the logical relation between $B11$ and $B12$ with respect to the upstream parent $A1$ is ''OR'' (Fig. 3(b)).

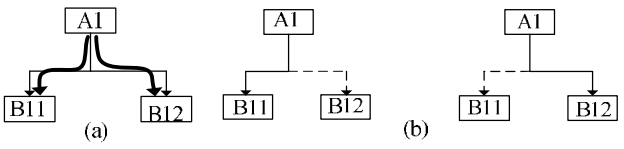


FIGURE 3. ''AND/OR'' logical relations between sibling nodes: (a) ''AND'' logical relation between B11 and B12;nd (b) ''OR'' logical relation between B11 and B12.

The logic relations between sibling components of the same parent can be assessed based on their parameter associations [32]. The parameter association between two

components C_i and C_j is expressed as

$$P(C_i, C_j) = \{p_{i1}, p_{i2}, \ldots, p_{ik}, \ldots, p_{im}\}, \tag{3}$$

where $P(C_i, C_j)$ denotes the set of parameters of C_i affecting the parameters of C_j; p_{ik} denotes a parameter element in the set; and m denotes element number in the set.

Using Eq. (3), the logic relation between components C_j and C_k corresponding to the common upstream component C_i is "AND" if $P(C_i, C_j) \cap P(C_i, C_k) \neq \emptyset$; otherwise, it is "OR." Note that C_j and C_k are sibling components of the same parent component C_i.

D. COMPONENT CHANGE BEHAVIOR

The change propagation process begins with a change in a single component but involves multiple components. The change propagation behavior can be divided into three categories.

1) CARRIER

When changes propagate through a carrier, it changes accordingly; however, it neither increases nor decreases the number of changes. In other words, changes input into a carrier induce changes in it; however, the same number of changes is output to its downstream nodes. A carrier does not increase the overall complexity of the change propagation problem.

2) MULTIPLIER

A multiplier can induce more changes in its downstream nodes than those propagated to it from its upstream nodes. The complexity of the change propagation increases because multipliers can stir up an "avalanche" of changes, exerting more negative effects on the budget or lead time of the product development.

3) ABSORBER

An absorber can absorb more changes than it generates. Thus, changes propagated from its upstream nodes can be absorbed and fewer changes are transferred to its downstream nodes. Absorbers reduce the overall complexity of the change propagation in the product development.

To understand and identify multipliers, absorbers, or carriers in a product, a reasonable posterior approximation of the CPI is introduced [33], [34]. In this index, a component i changes owing to a change propagating from other components connected with this component. The sum of links inducing changes from all other N components to i is denoted as $C_{in}(i)$, and the sum of change propagation links from the component i to all other components is denoted as $C_{out}(i)$. $CPI(i)$ characterizes the change propagation behavior in the absorber–multiplier spectrum and is expressed as

$$CPI(i) = \frac{C_{out}(i)}{C_{in}(i)}. \tag{4}$$

Clearly, a CPI value of greater than 1 indicates a multiplier component, whereas a value of less than 1 corresponds to an absorber component. Moreover, components with a CPI value of 1 are considered carriers.

E. PARAMETER CHANGE PROPAGATION MODELS

In the case of a CR in a product, the initial change component is selected by the product designer and the change gradually propagates to other components. Thus, the changes (if occur) in a component may be the initial ones or those caused by the direct or indirect change propagation from its upstream nodes. To facilitate the modeling of the parameter change propagation in a product, two variables are introduced: change probability "CP_{ij}" and change impact "CI_{ij}." The former is described as the probability that a component j must change based on an initial CR concerning component i. Alternatively, the latter is considered as the probability that the downstream successor component j changes when a change is propagating from its upstream predecessor i. Regarding the initial change component i, its propagation probability and change impact are expressed as CP_{0i} and CI_{0i}, respectively.

Fig. 4 presents a demonstrative example of the change propagation model. Three components C_i, C_j, and C_k in a product are linked by two directed edges ij and jk, indicated by bold lines with arrows in Fig. 4. The dotted line with an arrow that represents ik implies that C_i and C_k are not linked directly. A CR in the product entails initial changes in C_i and C_j. Only an initial change occurs in C_i because it has no upstream parents. However, C_i will transfer its change to C_j with CP_{ij} because they are directly connected, in addition to the initial change necessary for the CR. Regarding C_k, two sources of changes are possible. One is the change directly propagating from its upstream component C_j with CP_{jk}, and the other may indirectly originate from C_i with CP_{ik}.

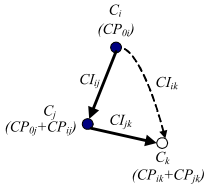


FIGURE 4. A demonstrative example for the change propagation model.

Each component along the change propagation path can undergo three types of changes: initial changes (if any), changes directly propagating from its upstream components, and changes directly propagating from its upstream components. Assuming a component k is linked to i ($i = 1$ to m) directly and to j ($j = 1$ to n) indirectly and CP_{0k} is the initial change probability (CP) of C_k, its overall CP OCP_k can be expressed as

$$OCP_k = CP_{0k} + CPI_i * \sum_{i}^{m} (CP_{ik} * CI_{ik}) + CPI_j * \sum_{j}^{n} (CP_{jk} * CI_{jk}), \tag{5}$$

where CPI_i denotes the CP index of C_i, indicating its role as a carrier, multiplier, or absorber; CP_{ik} represents the CP of C_k owing to the change propagating from i; and CI_{ik} represents

the change impact between C_i and C_k that are linked directly. Similarly, CPI_j represents the role of C_j; CP_{jk} denotes the CP of C_k for the change propagating from j; and CI_{ik} denotes the change impact between C_j and C_k that are linked indirectly.

An example of only three components C_a, C_b, and C_c is provided in Fig. 5. Using Eq. (5), the overall CPs of the components in Fig. 4 are

$$\begin{aligned} OCP_b &= CP_{0a} \\ OCP_b &= CP_{0b} + CPI_a \times CP_{ab} \times CI_{ab} \\ OCP_c &= CPI_b \times CP_{bc} \times CI_{bc} + CPI_a \\ &\quad \times CP_{bc} \times CI_{ac}. \end{aligned} \tag{6}$$

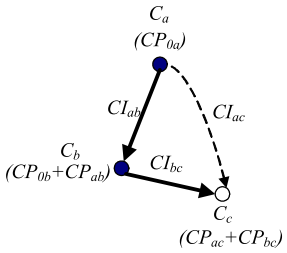


FIGURE 5. Three components C_a, C_b and C_c for change propagation model.

As discussed in Sec. I, a propagation path refers to a chain of linked components involved in the change propagation. Each component along the propagation path has a certain probability to change given by Eq. (5). Moreover, different change costs are involved in the case of different components selected to accomplish the CRs. Thus, the change cost for a change propagation path is the sum of the costs for each component participating in the propagation:

$$P_{pc} = \sum_{k=1}^{n} OCP_k * C_k, \tag{7}$$

where P_{pc} denotes the total cost of the change for each component along the change propagation pat;; n denotes the number of components along the change propagation path;; OCP_k represents the CP of component k (Eq. (5)); and C_k indicates its change cost. In this study, we aim to determine the optimal change propagation path with the minimum P_{pc} value in Eq. (7).

IV. NETWORK DESCRIPTION

A complex product embraces thousands of components linked with each other. Here, the components of a complex product are regarded as nodes and the links between them are regarded as edges. The components and their links constitute a complex network, where node weights are used to represent the CP caused by an initial change and edge weights are used to express the linking strength between components. The complex network model can be represented as

$$FDN = (V, E, I, w) \tag{8}$$

and

$$\begin{cases} V = \{v_1, v_2, \ldots\ldots, v_n\} \\ E = \{e_{ij} | i, j \in n\} \\ W = \{W(v_k) | v_k \in V\} \\ w = \{w(e_{ij}) | e_{ij} \in E\}, \end{cases} \tag{9}$$

where V denotes the set of nodes representing components connected with each other in a product, n denotes the number of total nodes, E represents the set of edges, e_{ij} represents the linking relation between nodes v_i and v_j; comma W denotes the set of node weights for V, $W(v_k)$ represents the weight for v_k; comma w denotes the set of weights for E, and $w(e_{ij})$ represents the linking strength between v_i and v_j. The node weight and edge weight of the network are described below.

1) NODE WEIGHT

The node weight $W(v_i)$ for node v_i is the CP required for the initial change. Obviously, $W(v_i)$ is equal to CP_{ij}. If a high CP is observed at the node, this node exhibits a high weight. Conversely, a tiny CP in the node indicates a low weight.

2) EDGE WEIGHT

The linking strength between two nodes can be expressed as the weight of the edge linking them. According to the literature [30], if a small change in the upper node can induce a change in the directly linked downstream node, the two nodes are linked tightly, indicating a large edge weight between them. Clearly, the edge weight $w(e_{ij})$ is equal to CI_{ij}.

V. CHANGE PROPAGATION PATH OPTIMIZATION ALGORITHM

A change propagation path is described as a finite sequence of components with dependencies between them [7]. It can be expressed as

$$CP = \{C_i, x_1, C_1, x_2, C_2, x_3, \ldots, C_m, x_m, \ldots, C_n, x_n, C_j\}, \tag{10}$$

where C_i represents the change-inducing component and C_j denotes the terminating one. Moreover, x_n represents the dependence between two directly connected components. Note that as iteration is common in a design process, C_m and C_n may be the same. For the sake of simplicity, an indirect dependence is beyond the consideration of this study.

This section presents an algorithm to determine the optimal change propagation path from all CPs expressed using Eq. (10). The algorithm evolves from the famous Dijkstra algorithm via several modifications to solve the optimal change propagation path problems.

First, we introduce the Dijkstra algorithm. Considering a weighted, directed graph $G = (V, E)$ with a source node s and weight function $w : E \rightarrow R$, the Dijkstra algorithm can be used to determine the shortest path from a single-source node by building a set of nodes at a minimum distance from the source [35]. A set S of nodes containing the final shortest-path weights from s is estimated and maintained. In other words,

for all nodes $v \in S$, $dist(v) = \delta(s, v)$ holds. This result is achieved by initializing three data structures [35].

1) ***dist***: This represents an array of distances from s to other nodes in the graph. The *dist* is initialized as $dist(s) = 0$, whereas for all other nodes v, $dist(v) = \infty$. As the algorithm proceeds, the *dist* is recalculated and finalized when the shortest distance to v is; chieved.

2) ***Q***: This represents the queue of all nodes in the graph at the start. When the algorithm is terminated, Q becomes empty;period

3) ***S***: This represents an initially empty set. As the algorithm runs, it stores nodes visited by the algorithm. S will contain all the nodes of the graph when the algorithm ends.

The algorithm proceeds using the following steps.

Step 1: When Q is not empty, insert v with the shortest distance $dist(v)$ in Q (not already in S). Notice that for $dist(s)$ initialized as *0*, s will be selected in the first run. In further runs, the node with the smallest *dist* value will be selected.

Step 2: Insert v in S, indicating that v has been visited and will never be visited again.

Step 3: Employ the adjacent nodes of the current node v to repeatedly update the *dist* values. For each new adjacent node u of v, if $\text{dist}(v)+weight(u, v) < dist(u)$, a new minimal distance is determined for u and $dist(u)$ is updated according to the new minimal distance value; otherwise, $dist(u)$ remains unchanged.

Step 4: Repeat steps 2 and 3 until all nodes in the graph are visited.

The Dijkstra algorithm can effectively solve single-source shortest-path problems with non-negative edge weights. For example, the Dijkstra algorithm is terminated at node d in Fig. 6(a) because no child nodes are linked with it and the shortest path a–b–d can be determined satisfactorily. However, when this algorithm is used to seek the optimal change propagation path, modifications are required. Let us consider the case shown in Fig. 6(b). The algorithm will seek two paths, a–b–c–e and a–b–c–d, and the path a–b–c–d will be identified as the shortest one. However, this is actually not the case. For node b that is a child of d, a change in d will inevitably propagate to b, thus forming one propagation loop. After several rounds, the sum of the costs for the change propagation along the loop path a–b–c–d–b–c–d–... will clearly exceed that along the path a–b–c–e. However, the Dijkstra algorithm cannot handle this issue, fundamentally because b has been visited as the child node of a and has been moved to S. It can never be visited again even though it is the child node of d and bears the change propagation from d.

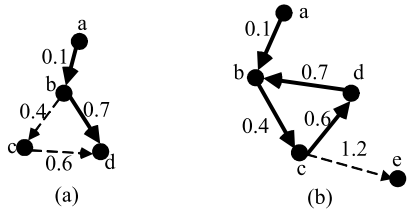


FIGURE 6. Demonstration of determining the optimal path using the Dijkstra algorithm: (a) satisfactory results;nd (b) unsatisfactory results.

Three modifications are made to the Dijkstra algorithm. First, when a node is visited, it is flagged instead of being inserted into S. Thus, a node may be visited several times if it is the child node of multiple nodes; consequently, loop cases can be considered. Second, only the downstream nodes of the current node v will be visited in the next run; hence, the change propagation paths will always be from the parent nodes to their child nodes. Third, the distance between the parent node and its child node is replaced by the value obtained using Eq. (5).

The demonstrative pseudocodes of the modified Dijkstra algorithm is shown below.

```
function Dijkstra(Graph, source)
  {
    dist[source] = 0 // Initially, the distance from source
to source is set to 0
    for each vertex v in Graph: // Initializations
      dist[v] = infinity //Unknown distance from the
source to each node set to infinity
      add v to Q// All nodes initially in Q
    end for
    while Q is not empty: // Main loop
    v = vertex in Q with min dist[v] // In the first run,
this vertex is the source node
    mark and remove v from Q
    for each child u of v : // Where u has not yet been
removed from Q
      if u is marked
        store the loop path
      else
      length(v, u)= value calculated with Eq. (5)
            alt = dist[v] + length(v, u)
      if alt < dist[u]: //A shorter path to u  has been
found
        dist[u]= alt // Update distance of u
    end for
    end while
    calculate total cost for each loop path and determine
the minimum one
  if the min cost − cost < dist[u]
    dist[u] = min cost − cost
  return dist[u]
  }
  end function
```

The most complicated parts of the modified Dijkstra algorithm are the nested ''while'' and ''for'' loops. For a network with n nodes and m edges, the ''while'' loop can be executed up to n times at most and the complexity of the ''for'' loop to handle the priority queue is O (log n). Thus, the overall time complexity of the modified Dijkstra algorithm is O ($n \times \log m$).

VI. CASE STUDY

In this section, we consider an example of an elevator system design. This example is used to demonstrate the practical applications of the proposed models and methods. Typically,

the core units of an elevator comprise the traction subsystem, guide subsystem, car and door subsystems, electrical control subsystem, and safety protection subsystem. Each subsystem includes many functional components interconnected with one another owing to dependencies. For brevity, only the safety protection subsystem is introduced in detail to describe the procedure for determining the optimal change propagation path. The safety protection subsystem comprises 15 components: traction wheel, traction motor, clutch, gear box, brake, reducer, rack and guide wheel, strainer, speed governor, traction machine, safety rope, traction rope, buffer, rope gripper, and safety gear. Table 1 presents the names and the corresponding indices of all the 15 components. Note that the change propagations attributed to indirect dependencies are neglected for simplicity.

TABLE 1. Components and their corresponding indices.

Index	Components	Index	Components	Index	Components
1	Traction wheel	6	Reducer	11	Safety rope
2	Traction motor	7	Rack and guide wheel	12	Traction rope
3	Clutch	8	Strainer	13	Buffer
4	Gear box	9	Speed governor	14	Rope gripper
5	Brake	10	Traction machine	15	Safety gear

A numerical DSM [7], [36] illustrates the direct dependency between the 15 components constituting the safety protection subsystem (Table 2). The information in the DSM is collected from the design and manufacturing datasheets, the repair and maintenance databases of the elevator system. For simplicity, the costs of the changes for each component are expressed in terms of the design completion time (days) and are listed in diagonal cells of Table 2. The numerals in off-diagonal cells denote CP_{ij} between i and j, with the row representing i indexes, while the column listing the j indexes. In other words, the changes propagate from the components in rows to those in columns. All indices in Table 2 match those in Table 1. For example, the first two cells with the bold numbers indicate that 2.2 days are required to handle the change in the traction motor (the first diagonal indexed by 1). Further, a change from the traction motor will cause a CP of 0.2 in the case of the brake component (the column indexed by 2).

Similarly, the numerical DSM in Table 3 shows CI_{ij} between C_i and C_j. Elements in the diagonal cells indicate the roles of the components (change carrier, multiplier, or absorber), and the numerals in the off-diagonal cells reveal CI_{ij}, with the direction from components in the rows indicated by the i indices to those in the columns denoted by the j indices. For instance, the first two cells with the bold numbers indicate that the traction motor (the first diagonal indexed by 1) is a change multiplier with a CPI of 1.6. Further, a change from the traction motor will be transmitted into a change impact of 0.2 with respect to the brake component (the column indexed by 2).

TABLE 2. Numerical DSM for CP_{ij} and costs.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	**2.2**	**0.2**		0.5			0.3				0.6				0.6
2		3		0.6				0.5				0.6			
3		0.7	1		0.4			0.6						0.3	
4			0.6	3.5											
5	0.3				2				0.7			0.3	0.2		
6			0.6			2.5				0.8			0.6		
7		0.7				0.6	2			0.5			0.3		
8					0.6			2			0.4			0.6	
9	0.4		0.8					0.8	4				0.4		
10										4.5					
11		0.6			0.6		0.7				2				
12	0.2											1.8			
13	0.5			0.3				0.6					1.7		0.8
14														1.6	
15															3

TABLE 3. Numerical DSM for CI_{ij} and component roles.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	**1.6**	**0.2**		0.7			0.5				0.2				0.3
2		1		0.7				0.2				0.4			
3		0.3	1		0.2			0.4						0.2	
4			0.6	0.7											
5	0.1				0.9				0.3			0.8	0.6		
6			0.2			2.5				0.2			0.6		
7		0.3				0.8	1			0.4			0.3		
8					0.5			0.8			0.3			0.6	
9	0.2		0.4					0.5	1				0.4		
10										1					
11		0.2			0.4		0.4				1				
12	0.3											1			
13	0.5			0.7				0.5					0.7		0.8
14														0.8	
15															2.6

Combining the data from Tables 1 and 2, the complex network visualized using Pajek toolkits (a set of tools for complex network visualization and simulation) [37] is demonstrated in Fig. 7. The node label corresponds to the node index, the design completion time (days), and its role (change carrier, multiplier, or absorber), while the line label denotes CP_{ij} and CI_{ij}.

Let us consider nodes 1 and 2 with their links (Fig. 8). As the arrow directs from node 1 to node 2, a change in the traction wheel (node 1) can evoke a change in the traction motor (node 2). However, a change in the traction motor (node 2) has no impact on the traction wheel (node 1). The node labeled "1 (Cost = 2.2, CPI = 1.6)" implies that handling changes in node 1 will cost 2.2 days, and this node is a change multiplier because its CPI exceeds 1. For node 2, the cost to handle the change will be 3 days and this node is a change carrier because its CPI equals 1. The line label between nodes 1 and 2 is reflected by CP_{12} and CI_{12}. It indicates that a change in node 1 will propagate to node 2 with a

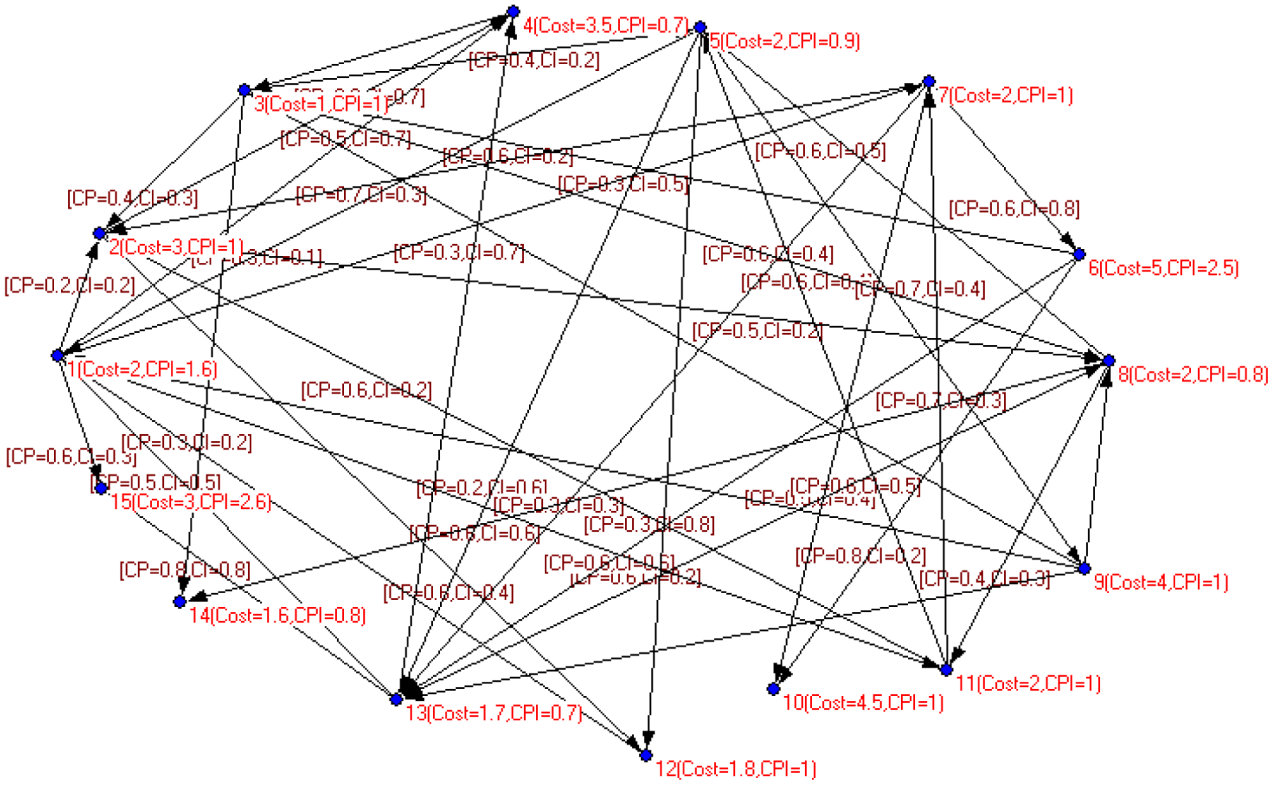


FIGURE 7. Network description for dependencies between the 15 components.

probability of 0.2. Correspondingly, node 2 must be modified with a probability of 0.2 because of the change propagated from node 1.

3) ONLY "OR" LOGICAL RELATIONS BETWEEN SIBLING COMPONENTS

Assume that the initial change occurs in node 1 (Fig. 8). We employ the modified Dijkstra algorithm to determine the optimal change propagation path among the 176 paths. If only "OR" logical relations exist between sibling components, no forks are possible along the path. In this case, the modified Dijkstra algorithm can be used to determine the optimal path of 1–2–8–14 (Fig. 9(a)). The total change cost for this path is

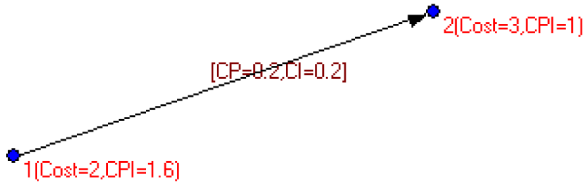


FIGURE 8. Detailed examination of nodes 1 and 2 with their links.

$$P_{pc}(1\text{–}2\text{–}8\text{–}14) = C_{k1} + CPI_1 \times CP_{1\text{–}2} \times CI_{1\text{–}2} \times C_{k2} + CPI_2 \times CP_{2\text{–}8} \times CI_{2\text{–}8} \times C_{k8} + CPI_8 \times CP_{8\text{–}14} \times CI_{8\text{–}14} \times C_{k14} = 2.2 + 1.6 \times 0.2 \times 0.2 \times 3 + 1 \times 0.5 \times 0.2 \times 2 + 0.8 \times 0.6 \times 0.6 \times 1.6 = 2.2 + 0.192 + 0.2 + 0.46 = 3.05 \text{ (days)}.$$

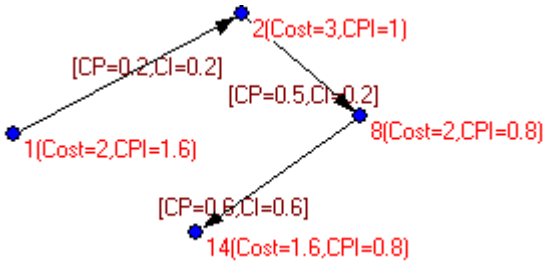
We investigate the DSM method introduced in [38] the result is 1–2–12, as shown in (Fig. 9(b)). The total change cost for this path using the DSM method is

$P_{pc}(1\text{–}2\text{–}12) = C_{k1} + CPI_1 \times CP_{1\text{–}2} \times CI_{1\text{–}2} \times C_{k2} + CPI_2 \times CP_{2\text{–}12} \times CI_{2\text{–}12} \times Ck_{12} = 2.2 + 0.192 + 1 \times 0.6 \times 0.4 \times 1.8 = 2.824$ (days).

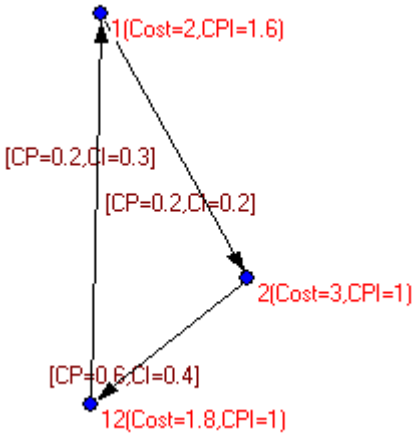
The path along 1–2–12 is preferable because its total change cost is less than that in the case of 1–2–8–14. However, an in-depth analysis reveals that a loop propagation path was neglected in the literature [38], which plays an important role. After the change propagates several runs along the loop, the total change cost will eventually surpass that of 1–2–8–14. Thus, the path 1–2–8–14 is the only optimal one.

4) "AND" LOGICAL RELATIONS BETWEEN SIBLING COMPONENTS

In this case, at least two sibling nodes of an upstream node should be simultaneously changed for the change propagated from their common parent node. We divide the optimal change propagation path search procedure into two steps. In the first step, the modified Dijkstra algorithm seeks the optimal paths from the initial node to the simultaneously changed sibling nodes. In the second step, the algorithm continues seeking the optimal paths from each individual changed sibling node to one terminating node that has no child node anymore.



(a)



(b)

FIGURE 9. (a) Optimal path obtained using the modified Dijkstra algorithm;nd (b) optimal path obtained using the DSM method.

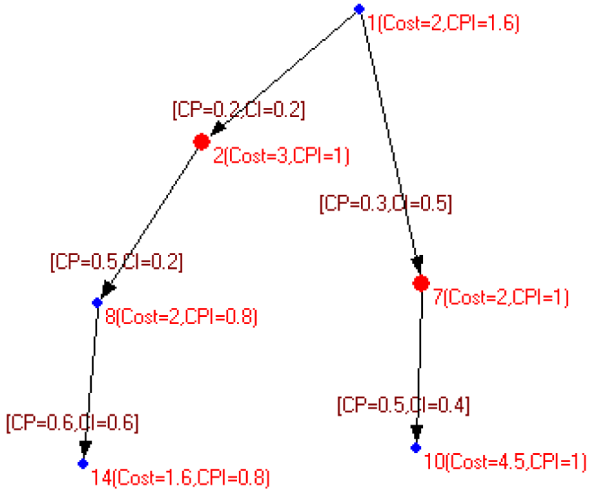


FIGURE 10. Optimal path depicting the "AND" logical relation.

Assuming that the initial change occurs in node 1 (Fig. 7), nodes 2 and 7 exhibit the "AND" logical relation and must change simultaneously as the child nodes of node 1. In this case, the modified Dijkstra algorithm is used to determine the optimal paths of 1–2–8–14 and 1–7–10 (Fig. 10).

The total change cost for this path is the sum of the two paths 1–2–8–14 and 1–7–10:

P_{pc} (2 AND 7) = P_{pc}(1–2–8–14)+Ppc(1–7–10) = 3.05+ 3.064 = 6.114 (days).

We performed a comprehensive investigation of past literature related to the applications of complex networks in the manufacturing field. We used the Web of Science, Scopus, Springer Link, IEEE Xplore, and other digital platforms related to the advanced manufacturing field. We found that the cases of "AND" logical relation between sibling components were rarely considered in the literature. The methods presented in this study were later appraised by the designers of Shangai Aoan elevator Co. Ltd. They appreciated the innovation of considering the loop change propagation cases and "AND" logical relations between sibling components. They showed satisfaction with the results and deemed the suggested methodology to be suitable for exploring the optimal change propagation paths during the product design and manufacturing process.

VII. CONCLUSION

This study introduces an advanced approach based on the complex network theory to determine the optimal change propagation path with the minimum cost to implement the change. Compared with the approaches proposed in previous studies, the proposed approach uses a modified Dijkstra algorithm combined with the complex network model to determine the optimal change propagation path. In addition to the role of components, such as change absorbers, carriers, or multipliers, the algorithm fully considers the loop change propagation paths and "AND" logical relations between sibling components. The proposed models and methods are tested by considering an industrial case example of an elevator system design. The results show that unlike the results obtained in the literature [38], the path 1–2–8–14 is the sole optimal path when the "OR" logical relation is considered between sibling components. Similarly, the "AND" logical relation are taken into account for the components can change simultaneously as the children of node 1 is 1–2–8–14 and 1–7–10 (Fig. 10). The results were deemed satisfactory by the elevator designers of Aoan Co. Ltd. The results of this study can help analyze different change propagation paths to determine the optimal path with minimum overall costs. Although the proposed models and methods are proved to be satisfactory, the future work along this line of research should include the following two points: (i) more realistic and accurate models, such as networks of networks (NONs), when depicting components and their dependencies are necessary and (ii) indirect dependencies that can propagate changes should be further considered because they are neglected in this study.

REFERENCES

[1] C. Eckert, P. J. Clarkson, and W. Zanker, "Change and customisation in complex engineering domains," *Res. Eng. Des.*, vol. 15, no. 1, pp. 1–21, Mar. 2004.

[2] H. Cheng and X. Chu, "A network-based assessment approach for change impacts on complex product," *J. Intell. Manuf.*, vol. 23, no. 4, pp. 1419–1431, 2012.

[3] S. Ma, Z. Jiang, W. Liu, and C. Huang, "Design property network-based change propagation prediction approach for mechanical product development," *Chin. J. Mech. Eng.*, vol. 30, no. 3, pp. 676–688, May 2017.

[4] C. M. Eckert, W. Zanker, and P. J. Clarkson, "Aspects of a better understanding of changes," in *Proc. Int. Conf. Eng. Design (ICED)*, Glasgow, U.K., Aug. 2001, pp. 147–154.

[5] P. Shankar, B. Morkos, and J. D. Summers, "Reasons for change propagation: A case study in an automotive OEM," *Res. Eng. Des.*, vol. 23, no. 4, pp. 291–303, Oct. 2012.

[6] T. Hu and Y.-Q. Fan, "Research on engineering changes based on PDM in aircraft project," *Group Technol. Prod. Mod.*, vol. 24, no. 1, pp. 19–23, 2007.

[7] I. Ullah, D. Tang, Q. Wang, and L. Yin, "Least risky change propagation path analysis in product design process," *Syst. Eng.*, vol. 20, no. 4, pp. 379–391, Jul. 2017.

[8] T. R. Browning, "Design structure matrix extensions and innovations: A survey and new opportunities," *IEEE Trans. Eng. Manag.*, vol. 63, no. 1, pp. 27–52, Feb. 2016.

[9] I. Gunawan, "Analysis of design structure matrix methods in design process improvement," *Int. J. Model. Simul.*, vol. 32, no. 2, pp. 95–103, 2012.

[10] B. Wang, F. Madani, X. Wang, L. Wang, and C. White, *Design Structure Matrix* (Innovation, Technology, and Knowledge Management). Cham, Switzerland: Springer, 2014.

[11] T. W. Simpson, "Product platform design and customization: Status and promise," *Artif. Intell. Eng. Des., Anal. Manuf.*, vol. 18, no. 1, pp. 3–20, Feb. 2004.

[12] N. P. Suh, *Axiomatic Design: Advances and Applications*. Oxford, U.K.: Oxford Univ. Press, 2001.

[13] M. F. Amro, "An engineering systems introduction to axiomatic design," in *Axiomatic Design in Large Systems*. Cham, Switzerland: Springer, 2016.

[14] X. Deng and W. Jiang, "An evidential axiomatic design approach for decision making using the evaluation of belief structure satisfaction to uncertain target values," *Int. J. Intell. Syst.*, vol. 33, no. 1, pp. 15–32, 2018.

[15] W. Fazar, "Program evaluation and review technique," *Amer. Statistician*, vol. 13, no. 2, pp. 646–669, 1959.

[16] R. J. Luttman, G. L. Laffel, and S. D. Pearson, "Using PERT/CPM (program evaluation and review technique/critical path method) to design and improve clinical processes," *Qual. Manage. Health Care*, vol. 3, no. 2, pp. 1–13, 1995.

[17] L. Mei, "Program evaluation and review technique (PERT) in construction risk analysis," *Appl. Mech. Mater.*, vols. 357–360, pp. 2334–2337, Aug. 2013.

[18] J. R. van Dorp, "A dependent project evaluation and review technique: A Bayesian network approach," *Eur. J. Oper. Res.*, vol. 280, no. 2, pp. 689–706, Jan. 2020.

[19] S. Sackey and B.-S. Kim, "Schedule risk analysis using a proposed modified variance and mean of the original program evaluation and review technique model," *KSCE J. Civil Eng.*, vol. 23, no. 4, pp. 1484–1492, Apr. 2019.

[20] G. A. Ollinger and T. F. Stahovich, "RedesignIT—A model-based tool for managing design changes," *J. Mech. Des.*, vol. 126, no. 2, pp. 208–216, Mar. 2004.

[21] J. C.-Y. Su, S.-J.-G. Chen, and L. Lin, "A structured approach to measuring functional dependency and sequencing of coupled tasks in engineering design," *Comput. Ind. Eng.*, vol. 45, no. 1, pp. 195–214, Jun. 2003.

[22] S. Li and L. Chen, "Identification of clusters and interfaces for supporting the implementation of change requests," *IEEE Trans. Eng. Manag.*, vol. 61, no. 2, pp. 323–335, May 2014.

[23] H. Lee, H. Seol, N. Sung, Y. S. Hong, and Y. Park, "An analytic network process approach to measuring design change impacts in modular products," *J. Eng. Des.*, vol. 21, no. 1, pp. 75–91, Feb. 2010.

[24] E. Pastor, J. Cortadella, and O. Roig, "Symbolic analysis of bounded Petri nets," *IEEE Trans. Comput.*, vol. 50, no. 5, pp. 432–448, May 2001.

[25] S. Balsamo, A. Marin, and I. Stojic, "Perfect sampling in stochastic Petri nets using decision diagrams," in *Proc. IEEE 23rd Int. Symp. Modeling, Anal., Simulation Comput. Telecommun. Syst.*, Oct. 2015, pp. 126–135.

[26] Y. Guodong, Y. Yu, Z. Xuefeng, and L. Chi, "Network-based analysis of requirement change in customized complex product development," *Int. J. Inf. Technol. Decis. Making*, vol. 16, no. 4, pp. 1125–1149, 2017.

[27] Z. W. Gong, H. C. Yang, R. Mo, and T. Chen, "Analysis of engineering change based on product development network," *Adv. Mater. Res.*, vols. 314–316, pp. 1607–1611, Aug. 2011.

[28] S. Ma, Z. Jiang, and W. Liu, "Evaluation of a design property network-based change propagation routing approach for mechanical product development," *Adv. Eng. Informat.*, vol. 30, no. 4, pp. 633–642, Oct. 2016.

[29] Y. Qin, L. Zhao, Y. Yao, and D. Xu, "Multistage machining processes variation propagation analysis based on machining processes weighted network performance," *Int. J. Adv. Manuf. Technol.*, vol. 55, nos. 5–8, pp. 487–499, 2011.

[30] M. V. Martin and K. Ishii, "Design for variety: Developing standardized and modularized product platform architectures," *Res. Eng. Des.*, vol. 13, no. 4, pp. 213–235, Nov. 2002.

[31] M. C. Pasqual and O. L. de Weck, "Multilayer network model for analysis and management of change propagation," *Res. Eng. Des.*, vol. 23, no. 4, pp. 305–328, Oct. 2012.

[32] D. Tang, I. Ullah, and L. Yin, *Matrix-Based Product Design and Change Management*. Singapore: Springer, 2018.

[33] M. Giffin, "Change propagation in large technical systems," in *System Design and Management Program*. Cambridge, MA, USA: MIT Press, 2007.

[34] E. S. Suh, O. L. de Weck, and D. Chang, "Flexible product platforms: Framework and case study," *Res. Eng. Des.*, vol. 18, no. 2, pp. 67–89, Aug. 2007.

[35] N. Abdullah and T. K. Hua, "Weighted methods of multi-criteria via Dijkstra's algorithm in network graph for less congestion, shorter distance and time travel in road traffic network," *Global J. Eng. Sci. Res.*, vol. 5, no. 7, pp. 46–57, 2018.

[36] Y. Li, W. Zhao, and X. Shao, "A process simulation based method for scheduling product design change propagation," *Adv. Eng. Inform.*, vol. 26, no. 3, pp. 529–538, 2012.

[37] A. Mrvar and V. Batagelj, "Analysis and visualization of large networks with program package Pajek," *Complex Adapt. Syst. Model.*, vol. 4, no. 1, pp. 1–8, Dec. 2016.

[38] D. Tang, R. Xu, J. Tang, and R. He, "Analysis of engineering change impacts based on design structure matrix," *J. Mech. Eng.*, vol. 46, pp. 154–161, 2010.

YONG YIN is currently a Professor with the Key Laboratory of Fiber Optic Sensing Technology and Information Processing, Ministry of Education, Wuhan University of Technology. His research interests include intelligent manufacturing, machine vision, and complex systems.

SHUXIN WANG is currently a Professor with the School of Intelligent Manufacturing and Electronic Engineering, Wenzhou University of Technology. His research interests include intelligent manufacturing, embedded control, and complex systems.

JIAN ZHOU is currently a Professor with the Key Laboratory of Fiber Optic Sensing Technology and Information Processing, Ministry of Education, Wuhan University of Technology. His research interests include intelligent manufacturing, computer communication, and complex systems.