Synthesis of feedback-based design concepts for sensors

Biplab Sarkar

Centre for Product Design and Manufacturing,
Indian Institute of Science, Bangalore, INDIA, 560012
biplab@cpdm.iisc.ernet.in

Amaresh Chakrabarti

Centre for Product Design and Manufacturing,
Indian Institute of Science, Bangalore, INDIA, 560012
ac123@cpdm.iisc.ernet.in

G. K. Ananthasuresh

Department of Mechanical Engineering,
Indian Institute of Science, Bangalore, INDIA, 560012
suresh@mecheng.iisc.ernet.in

Synthesis of feedback-based design concepts for sensors

We explore an approach to synthesize concepts of a class of sensors, where a quantity is sensed indirectly after nullifying its effect by using negative feedback. These sensors use negative feedback to increase the dynamic range of operation without compromising the sensitivity and resolution. The synthesis technique uses knowledge about existing phenomena to come up with an approach to synthesize concepts of sensors and also study their interactions with their surroundings, so as to generate robust designs. The approach uses a database of building blocks which are based on physical laws and effects that capture the transduction rules underlying the working principles of sensors. A simplified variant of the SAPPhIRE model of causality, which also uses physical laws and effects, has been adapted to represent the building blocks. SAPPhIRE model had been used earlier to understand analysis and synthesis of conceptual designs. We have adapted it here for automated generation of concepts. The novelty of the approach lies in the way and the ease with which it constructs a graph which is a super-set of the concept-space. The individual concepts are extracted out of the graph at a later point in time. The extraction of the concepts is done by using a modified breadth-first search algorithm which detects loops in the graph. The usage of breadth-first search algorithm for loop detection is novel, as we have demonstrated that it performs better than depth-first search algorithm for the specific problem. The technique has been implemented as a web-based application. For the sensor problems attempted, a number of existing patents were found that were based on the concepts that were generated by the synthesis algorithm, thus emphasizing the usefulness of the designs produced. The tool generated 35 concepts for accelerometers, out of which 2 concepts were found in patents. The synthesis approach also proposed new, feasible sensor concepts, thereby indicating its potential as a stimulator for enhancing creativity of designers. Automated generation of feedback-based sensor designs is a novel outcome of this approach.

Keywords: conceptual design; computational design synthesis; functional modelling;

indirect-sensing; closed loop;

1. Introduction

Market research predicts high demand for sensors in the near future (Wintergreen Research 2014; Freedonia Inc. 2015; Transparency Market Research 2015). To have a competitive edge, businesses need to be able to offer a variety of sensor designs and also push them to the market in the shortest possible time (George Stalk 1988; Thusu 2011). This research addresses this aspect by providing a support to assist the sensor designers for a fast generation of a variety of sensor design concepts.

Sensor designing process involves significant manual effort, along with past experience (Mukherjee and Fedder 1997; Antonsson and Cagan 2005). During the process of designing, the phase in which concepts are developed is called the conceptual design phase (Pahl and Beitz 1996). Researchers have observed that the number of concepts explored has a positive influence on the variety of solutions produced (Cavallucci 2002; Srinivasan and Chakrabarti 2009; Srinivasan and Chakrabarti 2010). However, manual exploration of a large concept-space is not possible. Various approaches for concept generation have been proposed by researchers. Computer-based conceptual design synthesis is an area of research that focuses on approaches to computationally support fast generation and exploration of the concept-space; a concept-space is a set of all concepts for the solutions of a problem. An overview of the computer-based design synthesis research is available in (Chakrabarti et al. 2011). In this research, computer-based design synthesis technique is used to achieve the above mentioned goals.

One approach that a number of researchers (Chakrabarti and Bligh 1996; Zavbi and Duhovnik 2000; Nagel et al. 2007; Srinivasan and Chakrabarti 2009) have adopted is to use physical laws and effects as building-blocks to generate concepts. This paper adapts this approach – building-block-based synthesis using physical effects – for conceptual designs of sensors. In particular, it focuses on a class of sensors that use negative feedback to increase the dynamic range of operation without compromising the sensitivity and resolution (Krishnan et al. 2012). Feedback sensors are an important class of sensors. According to market research the market share of feedback (i.e. closed-loop) sensors is increasing and will be more than that of direct (i.e. open-loop) sensors by 2020 (marketsandmarkets.com 2015). One such sensor is discussed next.

A force-balanced capacitive accelerometer, as shown in Fig. 1, is an example of a feedback-based sensor. Here, the acceleration induced displacement of the proof-mass, as shown in Fig. 2, is nullified by applying electrostatic force created by the balancing combs. The electrical voltage to the feedback combs is correlated to the acceleration to be measured. The feedback sensing mechanism uses the capacitance of the sensing combs to sense the displacement of the proof-mass. In contrast to the open-loop accelerometer where the sensitivity (mass/stiffness) and resonance frequency (square root of stiffness over mass) oppose each other, in the force-balanced accelerometer both sensitivity and resonance frequency can be independently tuned.

The specific objective of this paper is to develop a support for designers, to automatically generate concepts of feedback sensors and explore them. The intention is not to automate the entire design process, but to use the power of automation to improve the variety of the concepts explored by the designers at the early stage of the design process. Since loop is an inherent part in any feedback sensor (as shown in Fig. 2), loops play an important role in any feedback sensor synthesis algorithm.

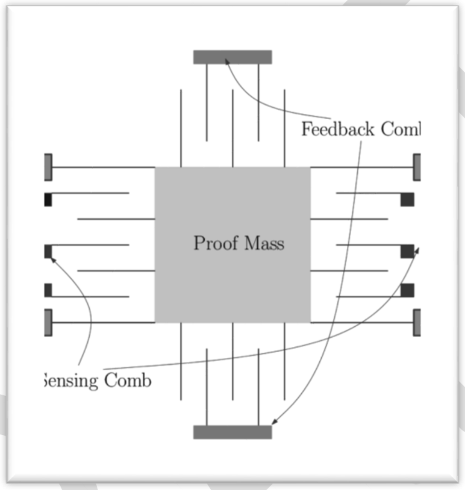

Fig. 1 Schematic of a force-balanced capacitive accelerometer

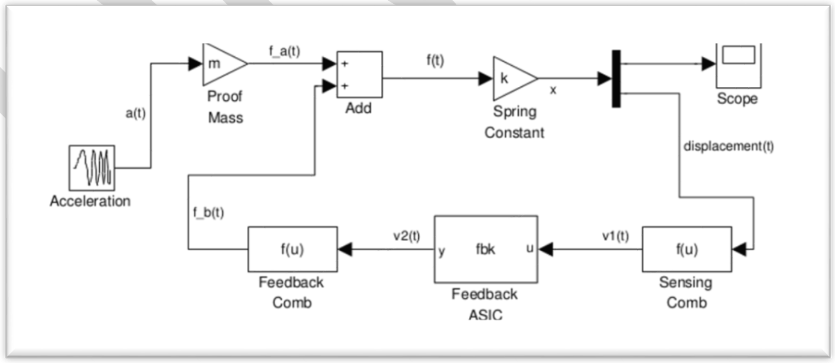

Fig. 2 Simulink (MathWorks Inc. 2015) representation of the block-diagram of the feedback principle

1.1. Related research

According to existing literature, computer-aided conceptual design research has adopted two broad approaches; one is the analogy-based approach and the other is the synthesis-based approach. Analogy-based approach can be classified into two categories: case-based (Goel et al. 1997; Watson and Perera 1997; Prabhakar and Goel 1998; Han and Lee 2006; Voss et al. 2012; Maher and Pu 2014) and bio-inspired (Chakrabarti et al. 2005; Vattam et al. 2008; Wilson et al. 2009; Nagel and Stone 2012; Goel et al. 2013) designs. In both the analogy-based approaches, new designs are created by using knowledge from past designs. Existing designs are altered to meet the design requirements of a new problem. If the existing design comes from a biological system, it is called bio-inspired design; case-based design primarily focuses on engineering systems as cases for adaptation. It is observed that solutions produced by using the analogy-based approach often have a bias towards familiar solution principles (Schmitt 1993) and as such, might lack variety, although this may not always be the case.

Synthesis-based approach can be classified into two categories: function-based (Chakrabarti and Bligh 1994; Malmqvist et al. 1996; Liu et al. 2000; Hirtz et al. 2002; Zhou et al. 2002; Bryant et al. 2005) and grammar-based (Hsiao and Chen 1997; Peysakhov and Regli 2003; Wojnarowski et al. 2006; Helms et al. 2009; Kurtoglu et al. 2010) synthesis. In function-based synthesis, a functional model is proposed, and then the model is used to develop solutions. In grammar-based synthesis, the focus is to develop a formal grammar with a set of rules that can then act on a design vocabulary to transform an initial design into a variety of new designs. The grammar is based on the state of the system (Schmidt et al. 2000; Campbell et al. 2009) and as such, may or may not have an underlying function or physical effect.

A number of researchers have used physical laws and effects as basic building blocks to develop functional models (Chakrabarti and Bligh 1996; Zavbi and Duhovnik 2000; Chakrabarti and Regno 2001; Zavbi and Rihtarsic 2010; Rihtarsic et al. 2012). This approach often generates solution principles with a greater variety, but occasionally ends up with a number of unrealistic solution principles. According to Chen et al. (2013) the reason for this is improper representation of physical quantities; Chen et al. (2013) have reported some progress in this direction by associating a flexible set of attributes to physical quantities.

In TRIZ methodology (Cavallucci 2002; Ilevbare et al. 2013; Yan et al. 2014), physical laws and effects were used in the form of a catalogue to support innovative thinking. However, they have not been used for automated generation of concepts. Campbell (Campbell et al. 1999; Campbell 2000) developed a procedure to synthesize electro-mechanical devices using agents and also facilitated their quantitative evaluation using a catalogue of standard components; however, no feedback-based designs were reported. Bond graph technique was used by researchers for concept generation (Bracewell et al. 1993; Bracewell et al. 2001; Wu et al. 2008). Since bond graph technique only deals with energy-based physical laws, it cannot support relations which are based on signals. Synthesis of sensors by using function-based building blocks have also been reported (Zhou et al. 2002). But they were much restricted to a given topology.

Various researchers have used graph grammars for synthesis (Starling and Shea 2005; Kurtoglu et al. 2006; Campbell et al. 2009; Helms et al. 2009; Kurtoglu et al. 2010; Koenigseder et al. 2015). They used grammar rules to generate new solution principles, but did not report any feedback-based concepts. In graph grammar technique, often a network is created with each node of the network representing a state or a concept (Koenigseder et al. 2015) and the links connecting these nodes representing the grammatical rules. The network represents the concepts space. The concept space may or may not be of finite size.

In all the reported cases, an exhaustive set of such solution principles or concepts would form the concept-space. However, generation of all solution principles is expensive, even though use of special algorithms such as bi-directional search makes it somewhat tolerable for smaller problems (Chakrabarti 2001). The approach presented in this paper differs significantly from all previous works because of its ability to construct a finite graph corresponding to the concept-space ahead of finding individual concepts. By constructing the concept-space ahead, finding a concept is much easier than using all other techniques as mentioned in the literature.

In this research we view sensors as a system that can be modelled as an input-output network. Among all the approaches that supports this view, we found SAPPhIRE (i.e. ***S**tate-change*, ***A**ction*, ***P**arts*, ***Ph**enomenon*, ***I**nputs*, *o**R**gans* and ***E**ffect*) (Chakrabarti et al. 2005) model of causality to be most appropriate for our research problem. SAPPhIRE uses physical laws and effects as building blocks (see Appendix-A for more details) and can be represented as an input-output network. Analysis and synthesis of conceptual designs can both be explained using this model (Srinivasan and Chakrabarti 2009). According to Chakrabarti et al. (2013), SAPPhIRE model can be used to capture different views of function within a generic model of design; function exists at the various outcome levels of abstraction of the SAPPhIRE model. Thus, different views of function expressed in models like function-behavior-structure (Gero and Neill 1998) and structure-behavior-function (Goel et al. 2009) also map into the SAPPhIRE model. Sensors are a class of devices that can be viewed as systems having functions that occupy the effect level of the SAPPhIRE model. Thus, synthesis of sensors by using SAPPhIRE model is possible. Although SAPPhIRE model is promising, it was never used before to describe or synthesize designs with feedback or to support quantitative analysis, where the magnitude of the changes produced by a solution principle could be estimated as part of synthesis. In order to support these, a simplified version of SAPPhIRE model called SAPPhIRE-lite was proposed (Sarkar et al. 2015a). In this research, we use this as the model for constructing the building blocks to synthesize conceptual designs. Details of these two models are given in Appendix-A and Appendix-B.

1.2. Proposed approach for synthesis

In this paper, a unique approach has been proposed to efficiently develop a comprehensive set of interesting (explained in section 6) solution principles with feedback. The technique takes an abstract view of the world in which phenomenon plays an important role. Phenomena cause interactions between components and thus change their states. If we take a holistic view of the system with its environment, we can better understand the possible desired and undesired interactions between them. The paper presents a way to capture the functioning of a system along with its interactions with the environment. In this research, we have constructed a graph of all interacting phenomena (explained in section-4.5) and have used it to synthesize sensor concepts. We have used this approach to design concepts of sensors and study their robustness against un-desired interactions from the environment. However, in this paper, the focus is on synthesizing feedback sensor concepts using this technique.

The proposed approach achieves this in five steps: (i) it proposes a representation for sensor concepts; (ii) it adapts a suitable model to capture the building blocks corresponding to the representation; (iii) it constructs a directed graph using the building blocks in linear-time; the graph corresponds to a super-set of concept-spaces for all types of sensors that we can generate; it also represents all interacting phenomena between a system and its surroundings; (iv) for a given specification of the intended sensor, it selects concepts, which are represented as paths in the directed graph, ranked with respect to a cost associated with the path; (v) it uses heuristic rules to avoid un-interesting concepts from being generated.

The use of heuristic rules to remove un-interesting solutions is an attempt to overcome the limitations of physical-law-based building-block approach as mentioned by Chen et al. (2013). The approach looks similar to graph grammar-based approaches, but it is far more constrained than graph-grammar approaches since it allows only one loop in a path, and all rules used are of physical relations; whereas, graph grammar rules may be defined for any physical or abstract state. In graph grammar each node represents a unique concept, whereas, a concept is mapped to a path in the proposed approach. The aim is to support designers of sensors to explore a large concept-space, thereby increasing their chances of developing solutions of greater novelty and utility.

In the following sections, we first present a representation for sensors, followed by models used to capture the representation in the form of building blocks; this is followed by a database of building blocks and the formation of a graph representing a super-set of concept-spaces for a number of sensors. The synthesis algorithm is presented next, followed by the validation and discussion. Finally, we conclude by summarizing the findings and the insights gained.

2. Representation

Sensor is a system which senses a quantity of interest without significantly disturbing it, and expresses the properties of the quantity sensed in terms of the properties of another quantity. We represent sensor using an input-output model. The property of the input quantity, which is of our interest, is measured in terms of the property of interest belonging to the output quantity. A sensor might have multiple inputs and outputs. For sensing, one of the inputs would be of the quantity to be sensed. During the process of sensing, all other inputs are assumed to be constant. If the other input quantities also change due to the change of the sensed input quantity, we get a feedback sensor. There can be more than one output quantity. We can choose a suitable output quantity. If we do not get the desired output quantity, we might try to cascade more than one sensor with different inputs and outputs to transform the output quantity to the desired one.

In the next section, we shall discuss about the model used to capture this representation of a sensor and use that to form building blocks for synthesis.

3. Model for building blocks

SAPPhIRE-lite, the model used for building blocks, has introduced four modifications to SAPPhIRE model: (a) the model is enhanced to focus on those areas that are relevant for computational sensor synthesis; (b) to capture complex scenarios, it concatenates multiple SAPPhIRE models; (c) it uses measurable quantities that undergo change in these scenarios; and (d) it makes a distinction between logical and quantitative attributes. The difference between the two models is explained in Appendix-C

The model is explained by using the example in **Table 1**. Here the input and output quantities are presented and their corresponding magnitudes are represented by using algebraic symbols (see lines 3-10). The output quantities are related to the input quantities by a relation (see line 1) that is mathematically expressed as an equation (see line 11). The condition for the activation of the relation is captured by using a set of predicates. The predicates are presented in lines 15-17. The condition here states that a solenoid is required for the relation to be active. Here solenoid is a conceptual structure (see line 14) that can be realized by using real components like a coil and air (see line 19). The state describes the observed changes of the parameters of the system (see line 20). The phenomenon is the interaction between the system and its surroundings. Here magnetic field and electrical current is exchanged between them (see lines 21-22). The overall intent would be to produce a magnetic field. This interpretation of the observable state change is captured as action (see line 23).

Table 1 A textual representation of a SAPPhIRE-lite instance

```
01    Relation: Ampere's Law
02    --------------------------
03    Inputs Quantities:
04      Length of solenoid (L)
05      Number of turns of solenoid (N)
06      Electric current (i)
07      Permeability (mu)
08
09    Output Quantities:
10      Magnetic Flux density (B)
11         Equation: B - iN/L * mu = 0
12
13    Condition:
14      Conceptual structure : Solenoid
15      Predicates:
16        Has Finite Electrical conductance = true
17        Has Air as Dielectric = true
18
19    Components: Coil, Air
20    States: No Magnetic Field present --> Magnetic Field present
21    Phenomenon:
22      Magnetic Field, Electrical Current
23    Action: Electrical Current produces magnetic field
```

4. Database

To aid in the synthesis process, we developed a database of building blocks. Known physical laws are modelled and populated in the database. The database consists of two portions: a *persistent* database and a *run-time* database. The *persistent* database is compact and is ideal for storage and transportation. It is used as a template to build the *run-time* database.

4.1. Persistent Database

The *persistent* database contains a set of models of known physical relations decomposed into sets of quantities, conditions, conditional attributes and relations. It captures information in textual form. For synthesis, we do not need all the entities of the model. So the *persistent* database is populated with a subset of these entities. In Fig. 3, we see a visual representation of the information stored in the *persistent* database. A relation contains reference to the input and output quantities. Each input and output quantity has an associated *symbol* for representing them in the mathematical equation for the relation. The equation is represented using the *eqn* key and is associated with the output quantity; there could be multiple outputs for a relation. Each relation has a *cost* attribute associated with it; it also has an associated condition.

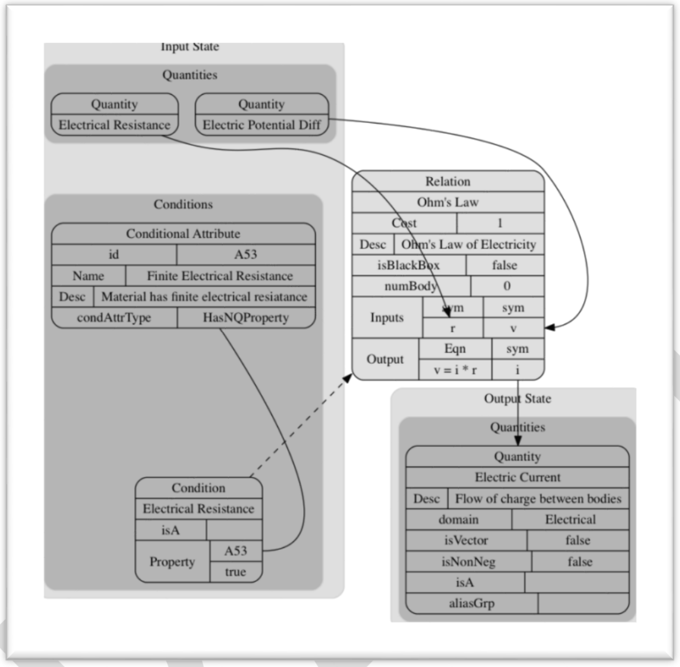

Fig. 3 Visualization of the data structures in the persistent database

A quantity may have an inheritance hierarchy; the attribute *isA* is used for this purpose. A quantity may also have multiple names; all of them are expressed as quantities and grouped together using the *aliasGrp* attribute. In the figure, we have shown the details for a single quantity; all quantities have similar structures. The quantities in the database are currently from multiple domains: mechanical (solid and fluid), electrical, magnetic, thermal, optical, chemical, radiation, acoustic, nuclear and general. For sensors domains play an important role and minimizing inter-domain transitions is recommended.

Condition is captured as a set of predicates using the *property* attribute; the name of the condition is used to describe a conceptual structure; conditions may exhibit inheritance hierarchy using the *isA* attribute. The persistence database uses textual representation to store this information. There are some more attributes to describe the quantity and its properties.

Cost is an implementation-specific attribute. The cost attribute of a relation is a way to rank concepts (see section 5.3). Optimizing the overall cost is a design goal. In order to facilitate that, typically, each relation has unity cost associated with it. To capture inheritance hierarchy among quantities, and to represent quantities with multiple synonyms, cost with value of zero is used, since these relations are reinterpretation of the same quantity (e.g. displacement resulting in a change of position).

4.2. Run-time Database

The model maps the output quantities to the input quantities with the help of some relations. Since both inputs and outputs are quantities, and a relation acts as a mapping function, we can see that there are two distinct sets here: a set is of quantities (SET_Q) and a set of relations (SET_R). A relation that belongs to SET_R, maps one or more input quantities to one or more output quantities. Also, a quantity may act as input for some relations and it can also act as output for some other relations. Hence, we see that between any two quantities of SET_Q there is no direct association; all associations are through an intermediate relation which belongs to SET_R. Similarly, there is no direct association between two relations belonging to SET_R, all associations involve an intermediate quantity that belongs to SET_Q. In graph-theory parlance, SET_Q and SET_R form a *bi-partite graph* if we use edges to represent the associations (see **Fig. 4**). The *run-time* database implements the *bi-partite graph* using data structures of the underlying computing language.

In the following sections, we shall explain how the model and the *run-time* database fits into a real system. We also interpret the meaning of the *run-time* database. The synthesis algorithm will also be introduced gradually over the next couple of sections.

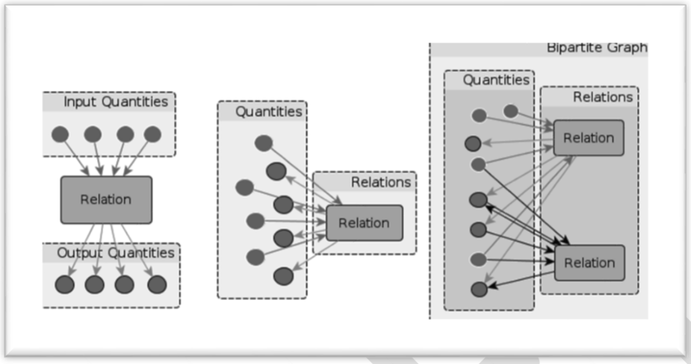

Fig. 4 Understanding the formation of the bi-partite graph in the database

4.3. Systemic view

A system is viewed as a hierarchical tree of components. Some component nodes of the tree are at the leaf level and others are not. A non-leaf node component has an associated sub-tree of components. We can see that at each of the non-leaf nodes of the tree, there exists some form of a relation that binds a child component node to its parent component node. Such type of relations can be expressed in the model. Hence, each component has in it some of the entities of the model. A closer look reveals that each node can have one or more models with entities like quantities from a node associated with entities like relations in another node. Thus it can be concluded that each component node has a subset of the *run-time* database object that we have introduced before.

A relation can have inputs that come from the components located at the same or different level with respect to itself; i.e., for a relation located at level *n*, the inputs can come from level *k*, where *k* can be *n*, less than *n*, or more than *n*. A vector quantity can flow across component boundaries. As such, for a relation that has input quantities that are vectors, there can be multiple variants of the relation to take care of the different combinations of the inputs coming from components of different levels. So, here we are introducing the need to have multiple variants of a relation in the *run-time* database.

4.4. Min-system

We have seen that in each component of the component tree, a subset of the *run-time* database exists. In this section we shall see how to build the *run-time* database for a system under design.

For synthesizing a system, the proposed synthesis algorithm starts with an initial size of the component tree. A system needs a minimum of three entities: surrounding, interconnection layer and a component; i.e. a component tree of three levels. We call this component tree of the system as the *min-system*. For building bigger systems, taller component trees may also be used. For the synthesis algorithm, the height of the component tree is a configurable parameter.

To build the *run-time* database, we populate all components of the *min-system* with the entities from the *persistent* database using directed edges to represent the direction of causal flow. This creates a bi-partite directed graph within the *min-system*. The generated graph is the desired *run-time* database for this *min-system*. While populating the relations in a component of the *min-system*, one has to see that the relation is compatible to that particular component as far as its required input and output quantities are concerned.

4.5. Super-set of Concept-spaces

Let us see what is the physical interpretation of the graph represented by the *run-time* database. The graph that is constructed inside the *run-time* database links all possible physical relations using directed edges. If we take a path that connects two quantities, we can see that the path has alternate nodes which are quantities and relations because of the *bi-partite* nature of the graph. So a path, in fact, signifies a set of transduction rules that would transform one quantity into another. This is, in fact, a conceptual design of a sensor. If we include all the possible paths between two quantities (say Q_a and Q_b), we shall get all possible concepts to sense the quantity Q_a in terms of Q_b. So the graph captures all possible concepts of a sensor, subject to the size of the database. Thus, the *concept-space* of a sensor is a sub-graph of this *bi-partite* graph. Since this graph can be used to synthesize many types of sensors, it is *a super-set* of many *concept-spaces* belonging to many sensors. We constructed this graph in linear time, i.e., it has time complexity of $O(n)$.

If we try to position this idea with respect to other concept exploration techniques, e.g. functional composition or graph grammar, we see that all other techniques have difficulty in exploring the concepts because of the presence of loops. They, generally, end up with a loop and hence the number of concepts are infinite and the same set of functional blocks or grammar rules are repeated indefinite times. In the later section, we shall use this global view to add constraints into our search algorithm and find concepts without running to infinity. Our approach has decoupled *concept-space* building activity from that of concept finding. Now to generate concepts, one has to look into the *concept-space* and select solutions instead of finding solutions. So, the goal of the synthesis algorithm is to find paths in this *bi-partite* graph that connects the input quantity to some desired quantity following a predefined topology.

In the next section, we will propose a synthesis algorithm that selects concepts from the *bi-partite* graph.

5. Concept generation

We have developed a tool (see Section-7), which can be used to generate two types of sensor concepts. For the first type, the user specifies both the input and the output quantities; the designs generated, referred to as *direct sensing* designs (Sarkar et al. 2015a), convert the input to the output. In the second type, the user only specifies the input quantity, and feedback is used to nullify the change caused by the input quantity that is being sensed; these are referred to as *feedback sensing* designs (Sarkar et al. 2015b). The focus of this paper is on supporting synthesis of *feedback sensing concepts.*

5.1. Feedback sensing designs

As mentioned in Section-1, all feedback sensors must have loops. In this type of designs, the user specifies the input quantity, and the algorithm searches through the *min-system* for all paths that lead to a feedback loop. It should be a negative feedback to make this solution practically useful. If the quantity that is fed back is a vector, the designer can easily create a configuration with negative feedback by reversing the direction; for scalar quantities, qualitative or quantitative analysis needs to be done to see if negative feedback would be possible. The structure of feedback sensing is shown in Fig. 5; the relations are represented using squares and quantities by ovals. The sensed quantity is of type *Q0* and has a magnitude *V*. After a series of transductions, the sensed quantity gets converted into type *Q1* and of magnitude *V0*. The magnitude of quantity *Q1* can be related to *Q0* using the relation *g(V)*. The feedback magnitude is *V1* and the resulting quantity is of magnitude *V2*. The feedback logic uses a series of transductions to generate the feedback quantity and is expressed by the relation *f(V2)*. If the input quantity *Q0* does not change during the time of the measurement, the feedback quantity might be able to catch up with the magnitude *V0* and nullify *V2*. At steady state, *V2* is null and $V1 = -V0$. The quantity *Q2* is controlled by feedback to indirectly track the input quantity *Q0*. So *V3* i.e. the magnitude of *Q2*, can be used to measure *Q0*.

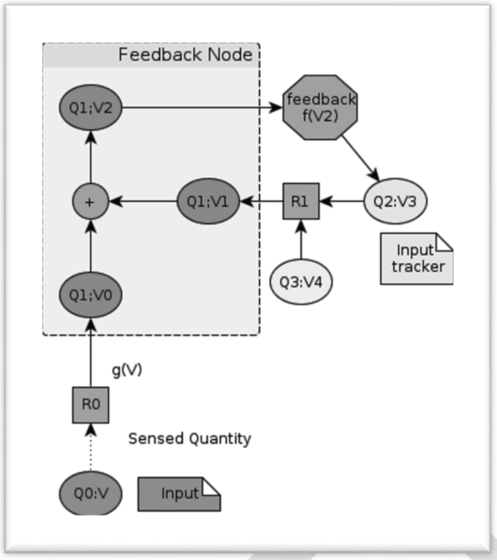

Fig. 5 Structure of a feedback sensor

5.2. Loop detection algorithm

In this section we describe the loop detection algorithm for feedback sensor synthesis. As discussed in Section-5.1, feedback sensors must have loops; loop detection is a critical part of this algorithm. Generally, *depth-first search* is the ideal algorithm for detecting loops exhaustively in a directed graph. In case of *depth-first search*, the child nodes are explored ahead of the peer nodes, and loops are detected by colouring the nodes traversed from the root, i.e. the start node, to the destination node. In such problems time is typically not a constraint since exhaustiveness is sought. However, loop detection problem in feedback sensor synthesis is different. Since the tool, of which synthesis is a part, is meant to be interactive for users to explore a variety of solutions as these get produced automatically by the synthesis algorithm, it is time, rather than exhaustiveness, that is the main constraint. So the synthesis algorithm should provide a relatively small number of concepts of high variety that can be explored in reasonable time by the user. *Depth-first search* is not practically feasible under such constraints when the graph is even moderately big. What is needed is a form of search that helps in discovering loops by exploring all alternative paths while also keeping the depth of the search to the minimum.

We used a variant of *breadth-first-search* algorithm called *breadth-first-search-loop-finder* which preserves the path information from the root node and helps in detecting loops. Thus, we imported the loop detection mechanism from *depth-first-search* into *breadth-first-search*. *Breadth-first search* is not the ideal algorithm for finding loops because of the space overheads. But we shall see in the subsequent sections that under added constraints of variety, path length and time, *breadth-first-search-loop-finder* algorithm performs better than *depth-first search*.

Table 2 shows the pseudo-code for the *breadth-first-search-loop-finder* algorithm. The algorithm uses a queue (see line 4) to explore all peer nodes ahead of exploring the children nodes. The path information corresponding to each node is saved in array attributes associated to the node but located in the queue (see lines 9-22). The algorithm scans all the edges and as such, may scan a node multiple times. So preserving path information outside the node is useful. The path information is propagated from the parent node to the child node when the child node is explored (see lines 36-47). In order to find loops, the nodes located in the path from the root is checked (see line 34). When a loop is found, the path information is returned (see line 52). The number of desired paths is also taken into account to stop the algorithm (see line 27 and line 50).

Table 2 Pseudo-code for breadth-first-search-loop-finder

```
01  // bfs with support for loop detection
02  function bfsLoopFinder (Graph, maxLoops) {
03      // Take a queue data structure
04      var a = Queue()
```

```
    // extract the root of the graph
    var m = Graph.root
    // initialize an array for storing the edges from
    // the root to the node
    var pathEdges = []
    // initialize an array for storing the nodes from
    // the root to the node
    var pathNodes = []
    // add the root node to the path of nodes
    pathNodes.push(m)
    // nodes information
    var nodeInfo = {
        'node': m,
        'pathNodes': pathNodes,
        'pathNodes' : pathNodes
    }
    // add the node information to the queue
    a.enqueue(nodeInfo)
    // list of paths to be returned
    loopPathEdges = []
    while(s = a.dequeue()) {
        // loop as long as there is some node in the queue
        if (maxLoops > 0){
            // restrict the number of loops what we are to find
            for each out bound edge e from s.node {
                // destination node for edge e
                var n = e.destination
                // we need to find loops. Check that the node
                // doesn't occur in the path
                if (s.pathNodes.indexOf(n) == -1) {
                    // preserve the path information
                    var pathEdges = s.pathEdges
                    pathEdges.push(e)
                    var pathNodes = s.pathNodes
                    pathNodes.push(n)
                    // nodes information
                    var nodeInfo = {
                        'node': m,
                        'pathNodes': pathNodes,
                        'pathNodes' : pathNodes
                    }
                    // preserve the node information in the queue
                    a.enqueue(nodeInfo)
                } else {
                    // we have a loop
                    maxLoops = maxLoops - 1
                    // return the path information
                    loopPathEdges.push(n.pathEdges)
                }
            }
        } else {
            break
        }
    }
    return loopPathEdges
}
```

The algorithm loops for all outgoing edges; its worst case complexity is dependent upon the type of graph and also on the number of solutions required. For a simple tree, the complexity is linear to the number of edges $O(|E|)$. But for arbitrary graphs, it has space complexity of $O(n^3)$ and time complexity of $O(n^2)$, which makes it worse than *depth-first-search* that has linear space complexity $O(n)$ and time complexity of $O(n^2)$. But for practical graphs, the complexity can be curtailed by limiting the number of concepts required. We shall see in the later sections that on average, *breadth-first-search-loop-finder* will produce shorter path based concepts much ahead of *depth-first-search*; thus the *breadth-first-search-loop-finder* algorithm will turn out to be much better for practical applications.

In the following sections, we will see some of the post processing activities associated with the set of concepts produced.

5.3. Ranking

Each generated concept is ranked with respect to the overall cost associated to its path in the *bipartite* graph. Each node in the graph has a cost attribute. Nodes which capture mapping between two quantities which are synonyms have a cost of zero; rest of them have a cost of one. The overall cost signifies the number of times energy transformation occurs and as each transformation is associated with some energy loss and some degradation of the signal, minimizing the overall cost is a design goal. So shorter paths are generally preferred. This technique gives a simple estimate about the quality of the concept produced without going deep into any quantitative analysis.

Once we filter concepts based on the path cost, we pass them through further filters that try to capture some of the rationale used by humans to develop a good concept. The next section discusses these filters.

6. Filtering of concepts

We assume a concept is *uninteresting* when one of the following occurs: (a) if there is repetition of relations or quantities in the solution principle; and (b) if there are too many domain transitions. Both the cases lead to inefficient designs, and therefore, are assumed to be *uninteresting* solutions. Limiting the number of *uninteresting* solution principles is a challenge for the synthesis approach.

We use filters based on heuristics to discard *uninteresting* solutions. We have adopted filters to avoid (a) duplicate relation, (b) duplicate quantity and (c) domain loop. Here, the scope of a relation is expanded by using a combined set of input and output quantities. With this approach, relations that are similar to one another but not exactly matching are also avoided. Also, by filtering out solution principles with duplicate quantities, long solution principles which might have unwanted loops can be avoided. Apart from these filters, the approach can support user-interactive filtering of concepts based on relations as additional constraints.

Often a feedback path involves some electronics to actively control the fed back signal and to interpret the measured quantity in terms of the sensed quantity. To facilitate this, there is an additional filter to select only solutions that have electrical voltage (*electric-potential-diff*) as a quantity in the feedback loop.

In the next section, we discuss a software implementation of the algorithm as a tool for supporting automated synthesis of feedback sensor concepts. The algorithm's complexity analysis is available in Appendix-E.

7. Software tool - SyCd

An interactive web-based tool, named as SyCd (***Sy**nthesis of **C**onceptual **d**esigns*) (Sarkar 2015), has been developed using the algorithms described earlier in this paper. It has a pictorial representation of the graph that shows a sensor concept in terms of the phenomena used in the transduction of its inputs to its outputs. The database currently has 169 relations and 127 quantities, which together captures 10 domains. The tool uses these technologies: HTML5, CSS, SVG, Javascript libraries (AngularJS, d3 and jQuery), MongoDb and web sockets. It has a server that stores the database of building blocks.

To generate a feedback sensor concept by using the tool, the user has to first enter the input quantity that is to be sensed. The tool then displays a number of possible concepts for the user to choose from. For easy comprehension, the search results are presented to the user after grouping them under three categories: domain view, relation view and conceptual-structure view. Each view divides the complete set of solutions into disjoint sub-sets. Domain-view is of importance for sensors. A solution principle might span across multiple domains. Decision making is easy when solutions are classified based on the number of domains. In relation-view, the solutions are grouped according to the order of the relations in the solution principle. This is useful when comparing a group of solution principles with an existing sensor concept. The conceptual-structure-view groups solution principles with respect to the set of conceptual structures that they use. This helps a designer to choose concepts based on conceptual structures. The user can also use relation-based filters for viewing only a specific type of concepts. The user can short list a handful of interesting concepts and then go to the analysis phase. In the analysis phase the user can investigate each concept and carry on detailed study of each concept with respect to other criteria. We shall not discuss about the analysis phase here as it is beyond the scope of the paper.

The process of populating the database has some difficulties. The database needs relations to capture the multiple views of the system. For example, to capture Coulomb's law of electrical force between charged bodies in each of its possible causal forms, one has to view the force from either of the charged-bodies. The other charged body may be assumed to be in the surroundings or embedded as a child component. So there has to be two instances of the law in the database so as to capture these two viewpoints. This is similar to the concept of *knowledge twisting* as mentioned in (Zavbi 2003; Zavbi and Rihtarsic 2010).

The next section presents a validation of the approach with respect to the quality of concepts produced.

8. Validation

Validation focuses on the evaluation of the quality of the solutions generated by the tool, based on four criteria: (a) comprehensiveness of the solutions produced along with quality, (b) practicality of the concepts generated, (c) novelty of the concepts, and (d) the time required to generate a set of concepts. The following sections provide details about the validation process.

To meet the comprehensiveness criteria, we need to see that the number of generated concepts are many, even after the un-interesting concepts are removed. For this, we can take the example of an accelerometer. The tool generated 35 concepts for accelerometer out of which patents were found based on 2 concepts. So, there are 33 concepts for which no patents were found. We argue that these 33 concepts should act as stimuli for the designer to develop sensors which will be new and interesting, thus enhancing the designer's creativity. Given the fact that 2 concepts of accelerometer were available as patents, we argue that this is a potential evidence that the concepts generated are practically relevant. We further argue that the remaining 33 concepts have potential to produce novel concepts of sensors as we could not see any existing patents for the same. In Section 8.1, we will see some of the concepts generated by the tool. In Appendix-C, we have an example to explain the process of comparing a patent with the generated concepts.

To validate that the set of concepts generated are fast enough to help the designer during the design process, we can see the performance of the algorithm in comparison with that of existing standard algorithms and see the superiority in the proposed approach. Section 8.2 details the results of such a comparison.

8.1. Results

In Fig. 6, concept of an accelerometer synthesized using this algorithm is shown. Acceleration is sensed by a mass that converts it to a body force. The force exerted causes stress, leading to strain. The strain is sensed by a strain gauge which uses an interferometer, thus causing some phase shift to a light wave. An optical detector is used to sense the standing waves that are produced due to the phase shift and gives an electric potential difference as output. The electric potential difference drives a phase shifter to nullify the interference patterns, thus creating a feedback. Here, the input acceleration can be measured in terms of the electrical potential difference. To drive the phase shifter, the electrical potential needs to be amplified; thus external electronic circuit is necessary to sustain the feedback loop. We found that US patent US4900918A has implemented this concept.

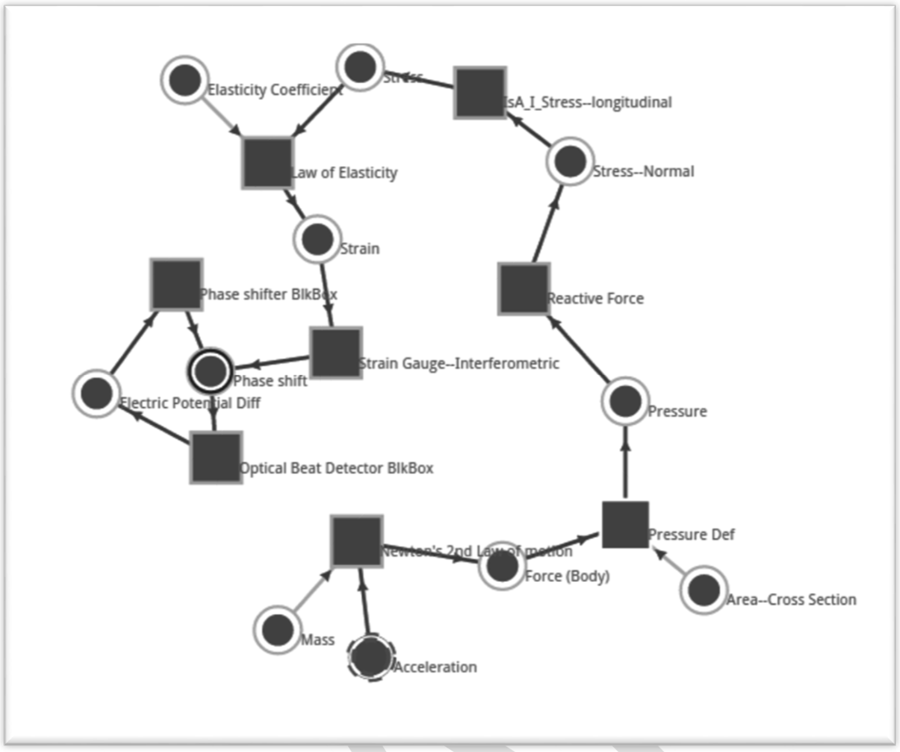

Fig. 6 Synthesised feedback sensor concept of an accelerometer that is similar to patent US4900918A

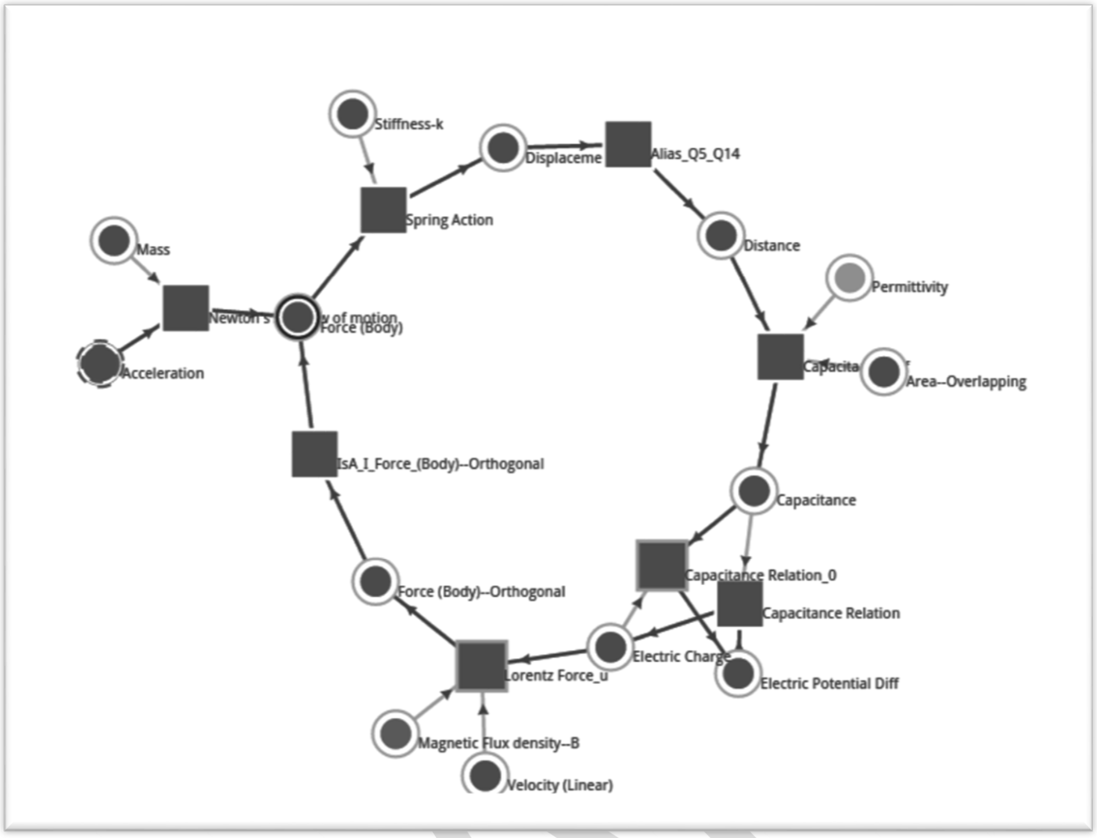

Fig. 7 Synthesised feedback sensor concept of an accelerometer that is similar to patent US6575029

Another example of an accelerometer concept that uses feedback is shown in Fig. 7. Here, the body experiences acceleration and exerts the resulting force on a spring, causing displacement. The displacement alters the value of position, in terms of a change of capacitance, in a position sensor that uses a capacitor. The changed capacitance produces an electric signal (*electric-potential-diff*) for an external feedback circuit to process. The external feedback circuit produces a feedback signal to nullify the force on the body by using Lorentz's force. We found that US patent US6757029 has implemented this concept.

In Fig. 8, the synthesised concept corresponding to the force-balanced accelerometer in Fig. 1 is presented. Here, the sensing path, from *force* to *electric-potential-diff* corresponds to the capacitive sensing comb and the path from *electric-potential-diff* to *force* corresponds to the capacitive feedback comb. The electronic control logic will be located at the *electric-potential-diff* node.

The synthesis tool produced 35 concepts for this accelerometer function. Patent search revealed existence of patents corresponding to 2 concepts.

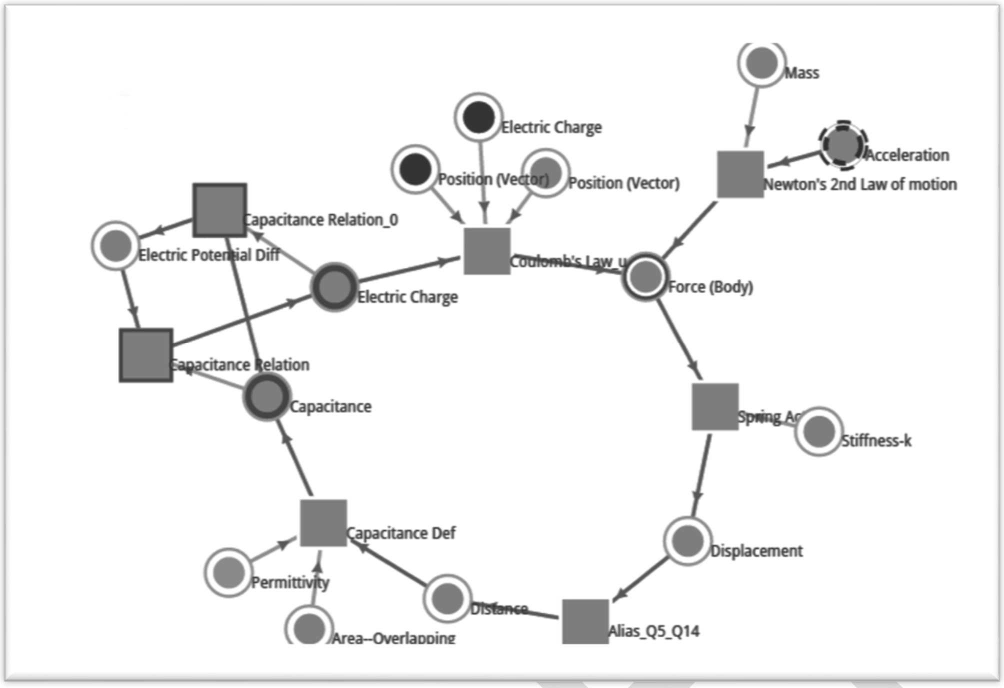

Fig. 8 Synthesised feedback sensor concept of an accelerometer with electronic feedback control

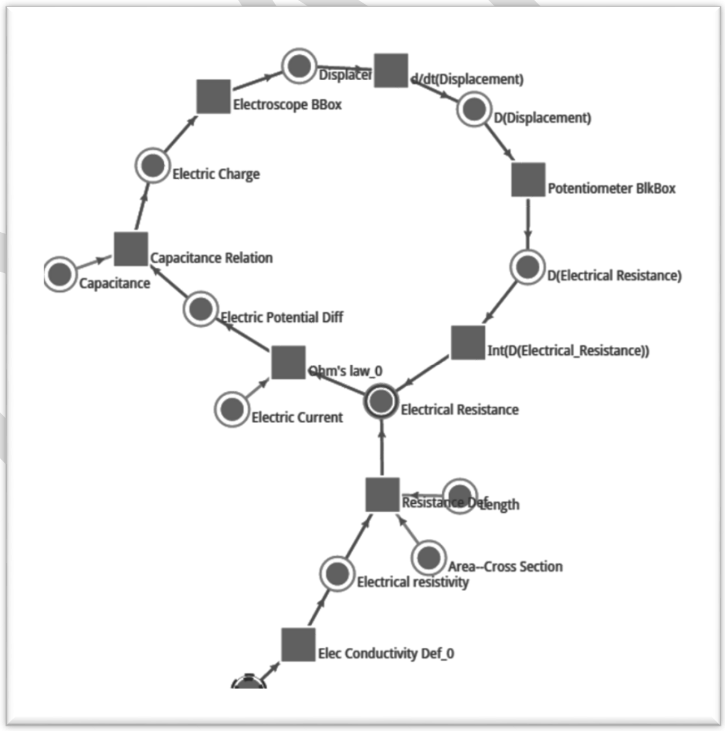

Fig. 9 Synthesised feedback sensor concept of a soil moisture sensor with electronic feedback control

In Fig. 9, a synthesized concept for a soil-moisture sensor is presented. The presence of moisture in soil alters its electrical conductivity, thus changing its electrical resistance. The change in the electrical resistance is sensed by passing a constant electrical current to produce an electrical potential signal; the electrical signal is passed to external feedback control circuit; the output of the feedback controller drives a capacitor by altering its electric charge, thus causing displacement to an attached gold-leaf electroscope;

the displacement of the leaves of the electroscope changes the resistance of the potentiometer (i.e. a variable resistor) which is formed by the position of the probes in the soil under test, thus nullifying the change in the electrical resistance of the soil. No closed-loop soil-moisture sensors designs have been reported yet in the literature (Jorapur et al. 2015) for us to compare with.

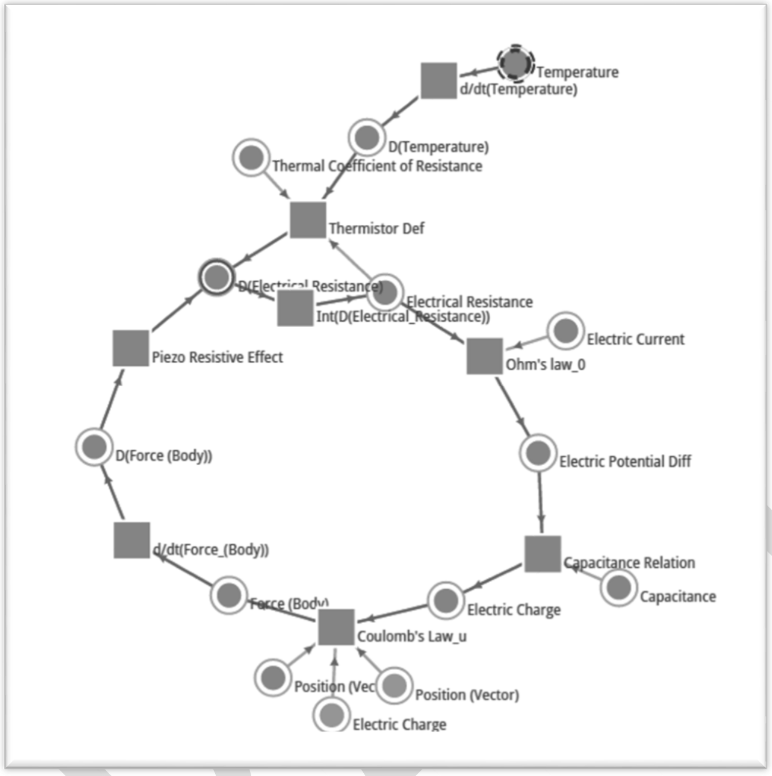

Fig. 10 Synthesised feedback sensor concept of a temperature sensor with electronic feedback control

In Fig. 10, a synthesised concept of a temperature sensor is presented. In this example, temperature alters the resistance of a resistor which is made with piezo-resistive material. The altered resistance is sensed by passing a constant electric current and the generated electrical potential signal is sent to an external feedback control circuit. The output of the feedback controller is used to drive a capacitive force generator to exert force on the piezo-resistor to alter its resistance, thus nullifying its effect. The existence of such a material is substantiated in (Tung Thanh Bui et al. 2009).

In the next section, we study the performance of the algorithm and the quality of the concepts produced.

8.2. Performance studies

We have measured the performance of the algorithm under various conditions to obtain insight into its usability and superiority.

In Fig. 11 we have investigated the dependence of the execution time and iteration on the number of concepts desired. The trend shows exponential in iterations and time with respect to the number of solutions. Since the software is run on a browser, the execution time is also dependent on external factors, like the system load; it is not possible to obtain memory usage information because of extensive use of application level cache, etc.

The performance of the *breadth-first-search-loop-finder* algorithm is compared with *depth-first search* to demonstrate its superiority under the application of the constraints. In Fig. 12, a comparison is made between *depth-first-search* and *breadth-first-search-loop-finder* algorithms in terms of the average length of solutions produced. As can be seen, when a given number of solutions are desired, the *breadth-first-search-loop-finder* algorithm performs better than *depth-first-search*; it produces much shorter solutions. It is not possible to generate and compare with the same set of solutions as *depth-first-search* needs to first generate all solutions and then sort out the solutions meeting the selection criteria.

In Fig. 13, the time difference for execution of *breadth-first-search-loop-finder* with *depth-first-search* is plotted for different number of solutions. No particular trend was observed but *breadth-first-search-loop-finder* takes more time for generating the same number of solutions.

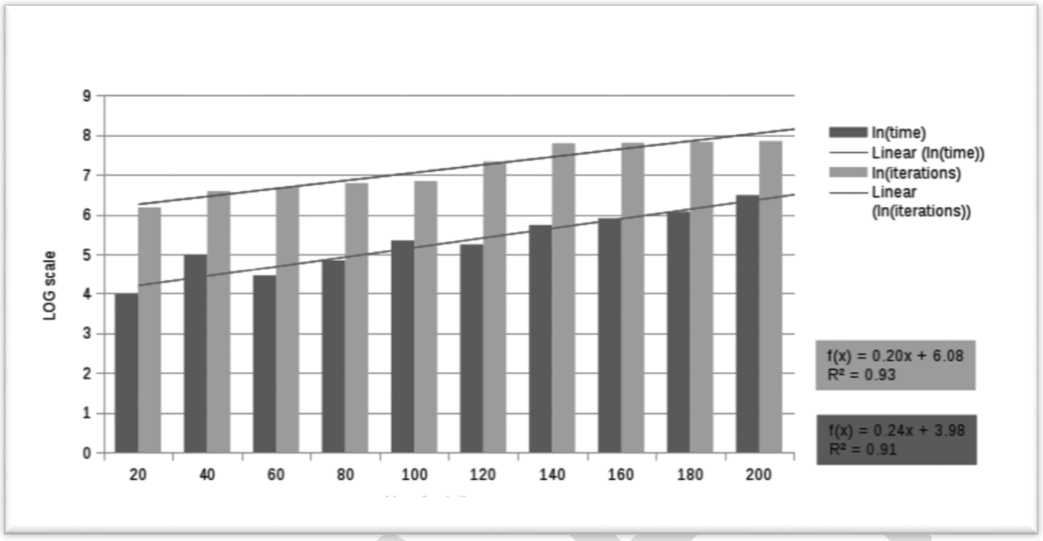

Fig. 11 Empirical study of the breadth-first-search-loop-finder algorithm

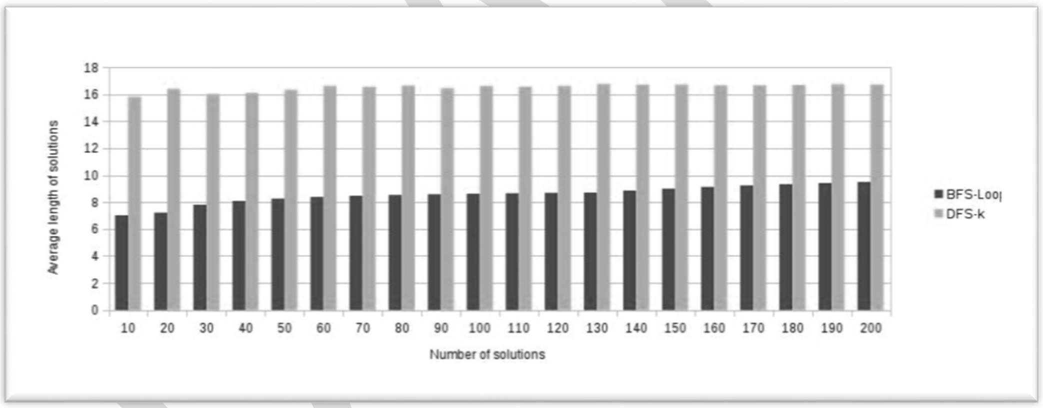

Fig. 12 Comparison between depth-first-search and breadth-first-search-loop-finder with respect to the average length of the solutions

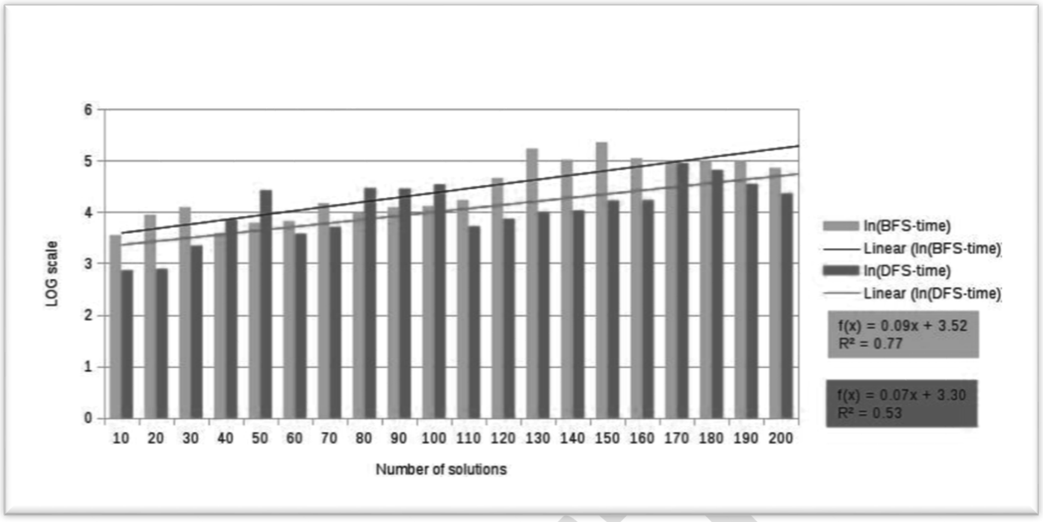

Fig. 13 Difference in time for same number of solutions

9. Discussion

In this section we discuss the results and their implications.

The exponential growth in Fig. 11 is because of the number of failed search attempts, and this corroborates with the understanding of the algorithm. The exact trend is not predictable as it depends upon the distribution of the desired solutions in the concept-space. The comparative advantage of the *breadth-first-search-loop-finder,* as shown in Fig. 12, supports the expected behaviour of more variety with shorter solution length. For the same number of solutions, even though *depth-first-search* produces much longer solutions, it might run faster than *breadth-first-search-loop-finder* as seen in Fig. 13.

The significance of the contribution is in the representation that makes it possible to generate feedback sensors in much the same way direct sensors are generated. Earlier work primarily employed the paradigm of constructing a sensor by concatenating building blocks one at a time, until it found the required output. Once the desired output is found, further construction would be seen as a wastage of effort and an inefficient approach, and therefore would not be encouraged. However, this is precisely what is required if a feedback sensor concept is to be generated: the same quantity must be produced at least twice in the path, where these two quantities must form a loop. What earlier work failed to see is the following. If individual sensors are seen as fragments of an underlying network of phenomena, the network would already contain the loops; therefore, the network would contain paths that represent direct sensor concepts, as well as paths that represent feedback sensor concepts. The representation used in this work, therefore, converts the synthesis problem into a graph traversal problem, where none of the sensor-possibilities are prematurely eliminated.

The other advantage that the representation brings is to make the synthesis process far more efficient. Since the network of phenomena is common across all possible sensors that can be constructed out of a given set of phenomena, the network needs to be constructed only once. The earlier, construction based paradigm would have to generate specific portions of the network over and over again, each time a new sensor had to be synthesized.

Automated generation of feedback sensor concepts is also unique. We have demonstrated a way to automatically synthesize feedback sensors. We have shown that the algorithm takes less time to generate quality solutions. The solutions generated were comprehensive enough to potentially enhance a designer's creativity. This allows the designer to explore a large concept-space in a short time. The fast exploration was possible because of the *breadth-first-loop-finder* algorithm. Since we have applied the approach for feedback sensors, and feedback sensors comprise a significant part of the sensor space (Ko 1996), this has the potential to extend the capability of the designer substantially.

The approach has a few limitations. Capturing existing relations with the building blocks is often a challenging task. Improperly captured relations give rise to solution principles that are difficult to visualize; so modifications are often required. The designer has to take care of the fact that only functional relations are currently captured by the model. For example, in case of a vibrating MEMS gyroscope, the vibrating body is intended to keep track of the inertial reference frame. But such interpretations cannot be represented by any input-output function. So such things cannot be easily modelled. But the angular shift in the orientation of the gyroscope with respect to the vibrating reference frame can be measured and hence can be captured as a function and thus can be modelled. Similarly, the effect of drag on the vibrating structure of the gyroscope can be modelled.

Even though the proposed model has a set of conditional predicates to deal with problems of incompatibility, the current software did not use them. There is no predicate processing mechanism in the current version of the software and as such, automatic avoidance of uninteresting solution principles is sometimes not possible. Often some of the relations are a function of scale, e.g. Van-der-Waals' force is effective when the distance is less than 10 nm. Such information is available in the conditional predicate and is not currently processed by the software. Improved implementation is required to alleviate the above limitations so as to explore the full potential of the model; this would be the focus of our future work.

The conditions associated with a relation are best evaluated during the embodiment design phase, since the real magnitude and dimension of the quantity is decided at that phase. Before this stage, all predictions are based on possibilities and as such may or may not happen in reality. Designers would have to use their experience and design catalogues to guess suitable magnitudes of quantities that would be used in concept evaluation.

It is to noted that this paper is primarily on exploration of concepts; therefore, the focus has been on functionality, and those phenomena that can be used to fulfil this functionality. However, efficacy of a sensor eventually lies not only in its phenomena but also on the way in which these phenomena are embodied, i.e. in the form and layout of the concept. The specific nature of the form and layout determines how well the phenomenon are embodied, and whether additional, unwanted phenomena, i.e. side-effects, can be adequately prevented. These aspects are currently out of scope of this paper. In order to provide a more comprehensive support to the development of sensors beyond the conceptual stage, these aspects need to be addressed.

The current software implementation did not explore the possibilities of predicting the characteristics of the components by using the set of predicates in the conditions associated with a model. Such an endeavour would also help in developing a proper component-level decomposition and thus aid in minimizing un-wanted effects. For a better description of the components, the geometric information can be captured by using qualitative techniques (Mukerjee and Joe 1990; Mukerjee 1991). Automated realization of components from a given prescription is a research problem. The class of problems that can be captured by the model is limited to those that have a functional relation between the inputs and the outputs. Thus, while selecting a problem to be solved by this technique, one has to think in terms of the effect caused by the change of one quantity on others.

After the concepts are generated and presented to the designer, the designer many accept some of the concepts and proceed to the next step in the design process, like quantitative analysis with some standard components or move into the embodiment design phase.

10. Summary

The paper presents a novel approach for automated generation of concepts for feedback-based sensor designs. The algorithm to produce such concepts is discussed. The approach presents a unique way to generate a *super-set of the concept-space* ahead of identifying the individual concepts. The algorithm uses a database that is based on the SAPPhIRE-lite model of causality. The paper also provides details of a software tool that uses the algorithm to synthesize solution principles for sensors. The paper uses a modified *breadth-first search* algorithm for detecting feedback. It presented a number of examples of sensor design principles generated by the tool that have implementation in existing patents, as well as, a set of solutions for which no existing patents were found. The approach can be applied for the synthesis of concepts for systems that involves effect level functions.

11. Acknowledgments

The authors would gratefully acknowledge the financial support from the National Programme on Micro and Smart Materials and Systems (NPMASS), Government of India for this work.

12. Appendices

Appendix-A. SAPPhIRE Model

We shall summarize the SAPPhIRE (Chakrabarti et al. 2005) model of causality in this section. The acronym SAPPhIRE stands for ***S**tate-change*, ***A**ction*, ***P**arts*, ***Ph**enomenon*, ***I**nputs*, *o**R**gans* and ***E**ffect*, which are the seven constructs of the model. SAPPhIRE introduces these seven levels of abstraction to capture causality in physical systems. At the lowest level, it has parts which have organs (i.e. properties) that, along with external disturbances, serve as inputs to activate physical effects, thus causing some observable phenomenon in the form of an exchange of material and energy between the system and its surroundings, leading to changes to the state of the system. . The change of state may be interpreted as an action. The change of state many also act as an input into another SAPPhIRE model. Thus multiple SAPPhIRE models may be connected in chains to capture the functioning of more complex physical systems.

Appendix-B. SAPPhIRE-lite Model

In this section we shall summarize the SAPPhIRE-lite (Sarkar et al. 2015a) model of causality. SAPPhIRE-lite focuses mainly on the input, organ and effect layers of the SAPPhIRE model. It alters and generalizes them to describe complex scenarios by which sensors work, including those involving feedback. It has changed the nomenclature and scope of some of the entities of the SAPPhIRE model to support quantification and to capture feedback. It uses a variant of Signal Flow Graph called *Switching Flow Graph (SFG)* (Smedley and Cuk 1994) to quantitatively relate input quantities to output quantities under favourable input conditions. SFG is an input-output relation that is controlled by logic; only when the desired logical conditions are met, the output is produced as a function of the inputs. SFG is used to capture relationships among attributes without considering as to how the relationships are achieved or maintained.

The block diagram of the SAPPhIRE-lite model, as shown in Fig. 14. It has retained SAPPhIRE's levels of abstraction. It captures the state of the system over a span of time. It has explicitly depicted the states and presented them as an input-output system. The state of the system is described by a set of measurable quantities and conditions belonging to the underlying level of components. Physical phenomena are the underlying exchanges of material, energy or signal between the system and its surroundings that take place because of the activation of various physical effects. These exchanges in turn lead to state-changes. These state-changes are interpreted as actions. Physical effects get activated when favourable input conditions are met and required inputs are available. A system's causal behaviour can be described using this model as a pictorial graph. Mathematical analysis of its state variables is also possible. The various elements of the model are summarized next using the example in Table 1.

Quantity: Measurable material properties and physical attributes are represented as quantities; e.g. Input and Output quantities in Table 1(lines 3-10).

Conceptual Structure: The underlying components (explained later) are abstracted using conceptual structures (Chakrabarti 2004; Rihtarsic et al. 2012). Conceptual structures have quantities. An example can be seen in Table 1 (line 14).

Condition: A set of logical predicates forms a condition. Information about the context or the situation for a relation (explained later) to take place is described using conditions. A condition is named after a conceptual structure. For example, see in Table 1 (lines 16-17).

State: The quantities and the conditions of the underlying components define the state of a system. It is an abstract entity. Because quantity is time-dependent, the state of a system is also time-dependent. An example of state change can be seen in Table 1 (line 20).

Relation: The input quantities are mapped to the output quantities using a relation. Each output quantity can be mathematically related to the set of input quantities. This makes it possible for SFG to be used for modelling relations. A relation may have an associated condition for it to get activated. For example, the relation captures Ampere's law and associates the input quantities to the outputs quantity using a mathematical relation (i.e., expressed using Latex (2015) notations) as mentioned in Table 1 (line 11).

Component: Components are physical embodiment of the conceptual structure. Components may contain other components. For the example in Table 1 (line 19), a coil captures the functionality of a solenoid and air as a dielectric material.

Action: The change of state exhibited by the system is interpreted as an action. In the example in Table 1 (line 23), one can interpret this state change as an action performed by the system to generate a magnetic field. It can be also interpreted as a way to change the magnetic field in a region.

Phenomenon: A phenomenon is an interpretation of the exchange of material, energy and signal between a system and its surroundings; e.g. see Table 1 (line 22). It is a result of physical effects that is taking place. Since SAPPhIRE-lite is based on signal flow and does not *explicitly* capture the flow of material or energy, it only indirectly captures phenomena. The exchange of material and energy is captured in the form of inputs or outputs.

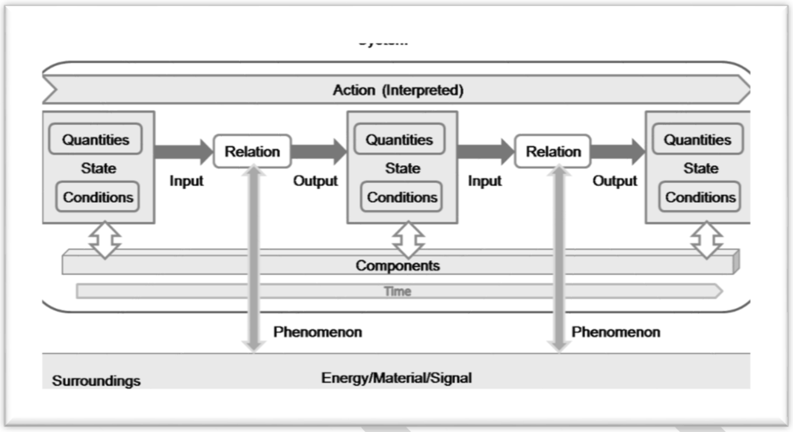

Fig. 14 Pictorial description of the SAPPhIRE-lite model

Appendix-C. Difference between SAPPhIRE and SAPPhIRE-lite

The difference between SAPPhIRE and SAPPhIRE-lite is shown in **Table 3**. SAPPhIRE-lite's quantity is of numeric type. In SAPPhIRE model, input is of numeric type and organ can either be of numeric or boolean type. SAPPhIRE's numeric entities, i.e., input and organ of numeric type, are mapped as quantity in SAPPhIRE-lite model. Boolean type of organ in SAPPhIRE is mapped as condition in SAPPhIRE-lite. A state in SAPPhIRE maps as a state of SAPPhIRE-lite. SAPPhIRE's effect is generalized into a relation in SAPPhIRE-lite. The link between effect with its inputs and its organs is replaced with signals in case of SAPPhIRE-lite. Phenomenon is explicitly an exchange of quantity in SAPPhIRE-lite, whereas, it was implicitly defined as an exchange in SAPPhIRE. Parts in SAPPhIRE used to be the smallest physical entity; in SAPPhIRE-lite a component is used instead; components can be broken down into other smaller components. There is an explicit concept of output in SAPPhIRE-lite, whereas, output was implicit in SAPPhIRE. Abstraction of components in the form of a conceptual structure is unique to SAPPhIRE-lite. SAPPhIRE-lite is focused at formalizing SAPPhIRE model so as to take into account computational aspects for it to be used for sensor synthesis.

SAPPhIRE had been used earlier for modelling relatively simple systems involving single phenomenon. SAPPhIRE-lite has been developed to model more complex systems that involve use of multiple phenomena. In the next section, we shall see how the SAPPhIRE-lite model can be used to create a database of building blocks to be used for synthesis of such systems.

Table 3 Difference between SAPPhIRE and SAPPhIRE-lite

SAPPhIRE		SAPPhIRE-lite	
Input	Numeric	Numeric	Quantity
Organ	Numeric	Numeric	
	Boolean	Boolean	Condition
State	Input	Quantity	State
	Organ	Condition	
Effect	Link	Signal	Relation
	State	State	
Phenomena	Exchange	Exchange of Quantity	Phenomena
Parts		Recursive	Components

		Condition name	Conceptual Structure
Output	Implicit	Explicit	Output

Appendix-D. Patent-based validation

In this section we will discuss about the process of validating a generated concept such as the one in Fig. 6 with that of an existing patent e.g. US 4900918 A.

The patent US 4900918 A is titled: *Resonant fiber optic accelerometer with noise reduction using closed loop feedback to vary path-length.*

The abstract says:

A pair of resonant optical cavities are used to measure shifts in length of a beam which deflects under acceleration. One end of each optical cavity is highly reflective and the other end partially reflective, thus permitting resonating light to exit the resonators. The light outputs of the two resonators is combined and the resultant *beat frequency is used to detect deflections of the beam.*

So we see a change in length is measured with optical resonators using beat frequencies. Inside the body of the patent description we see how the change in length happens.

With the application of an acceleration, the beam becomes bent out of shape and the optical path-lengths become different, and thus there is a different resonant frequency associated with each.

So the change in length is because of acceleration. The feedback part is described in the following text.

The resonant accelerometer can also be used in the closed loop feedback mode. In this case, a phase shifter can be included in one of the waveguides. Instead of using an oscillating signal on the shifter, a dc signal will be used. The phase shifter can vary the optical path-length of one of the waveguides, and therefore can be used to equalize the two path-lengths which have been made unequal by the acceleration input. A unique voltage is required to vary the index in the phase shifter region so that no beat frequency appears at the output. A feedback loop may be constructed to achieve this condition, and the voltage to the shifter will be a measure of the acceleration. In the linear region of the shifter, the output voltage will be directly proportional to the optical path-length difference.

Thus the idea matches to that of the concept that we have generated in Fig. 6. This is how we validated the concepts with that in existing patents.

Appendix-E. Analysis of the synthesis algorithm

The complexity analysis of the synthesis algorithm is presented here.

Definition-1: The length of a solution with a feedback is the maximum length between the start node and the node with a feedback.

Theorem-1: The algorithm breadth-first-search-loop-finder is exhaustive.

Proof: The exhaustiveness of the algorithm is in comparison with the solutions generated by a *depth-first-search* algorithm on the entire graph. The set of any *k* shortest solutions will have to be same for both the algorithms in order to satisfy the exhaustiveness criterion.

Let us use contradiction to prove the theorem. Let us assume that there is a path *j* which is one of the best *k*-paths that connects the start node and ends up into a feedback loop and is *not* in the solution set of the *breadth-first-search-loop-finder* algorithm. To make this possible there has to be some edge, say *e*, and node say *dst(e)*, in the path *j* that is ignored by the *breadth-first-search-loop-finder* algorithm. Assuming that the graph has no duplicate edges, all edges and the associated destination node will be pushed into the queue. So, edge *e* and its destination node *dst(e)* is also in the queue; thus contradicting the initial assumption ■

Theorem-2: The algorithm *breadth-first-search-loop-finder* has a worst case time complexity of $O(n^2)$. Assume *n* to be the number of nodes in the graph. Also assume that the algorithm should find out all the possible solutions.

Lemma-2.1: Maximum number of push into *queue* is $O(n^2)$.

Proof: The algorithm runs as long as there are some entries in the *queue* (line-25). For each of the outgoing edges a node gets pushed into the *queue* (line-47). Since there can be $O(n^2)$ edges, the maximum number of pushes will be $O(n^2)$. ■

Using *Lemma-2.1*, it is trivial to see that the over all time complexity is in the order of $O(n^2)$. ■

Theorem-3: The algorithm breadth-first-search-loop-finder has space complexity of $O(n^3)$.
Proof: According to the *Lemma-2.1*, the number entries that is pushed into the *queue* is $O(n^2)$. Each time when an entry gets pushed, an array of nodes that represents the path information from the root node also gets pushed. Since there are n-nodes, the size of a path is $O(n)$. So the memory requirement is $O(n^3)$.■

13. References

Antonsson EK, Cagan J (2005) Formal engineering design synthesis. Cambridge University Press

Bracewell RH, Bradley DA, Chaplin RV, et al (1993) Schemebuilder: A design aid for the conceptual stages of product design. In: International Conference on Engineering Design, IECD'93. Citeseer,

Bracewell RH, Shea K, Langdon PM, et al (2001) A methodology for computational design tool research. Proc ICED01 Glasg Scotl Vol"Design Res 181–188.

Bryant CR, Stone RB, McAdams DA, et al (2005) Concept generation from the functional basis of design. In: Int. Conf. on Eng. Design. Engineers Australia, Melbourne,

Campbell MI (2000) The A-Design invention machine: a means of automating and investigating conceptual design. Carnegie Mellon University

Campbell MI, Cagan J, Kotovsky K (1999) A-design: An agent-based approach to conceptual design in a dynamic environment. Res Eng Des 11:172–192.

Campbell MI, Rai R, Kurtoglu T (2009) A stochastic graph grammar algorithm for interactive search. In: Proc. IDETC/CIE Conf. ASME, pp 829–840

Cavallucci D (2002) TRIZ, the Altshullerian innovation approach to solving problems. In: Chakrabarti A (ed) Engineering Design Synthesis. Springer London, pp 131–149

Chakrabarti A (2001) Improving efficiency of procedures for compositional synthesis by using bidirectional search. AI EDAM 15:67–80.

Chakrabarti A (2004) A new approach to structure sharing. J Comput Inf Sci Eng 4:11–19.

Chakrabarti A, Bligh TP (1994) An Approach to Functional Synthesis of Solutions in Mechanical Conceptual Design. Part I: Introduction and Knowledge Representation. Res Eng Des 6:127–141.

Chakrabarti A, Bligh TP (1996) An approach to functional synthesis of mechanical design concepts: theory, applications, and emerging research issues. AI EDAM 10:313–331.

Chakrabarti A, Regno R (2001) A new approach to structure sharing. In: Proc. Intl. Conf. on Eng. Design. pp 155–162

Chakrabarti A, Sarkar P, Leelavathamma B, Nataraju BS (2005) A functional representation for aiding biomimetic and artificial inspiration of new ideas. AI EDAM 19:113–132.

Chakrabarti A, Shea K, Stone R, et al (2011) Computer-based design synthesis research: an overview. J Comput Inf Sci Eng 11:021003.

Chakrabarti A, Srinivasan V, Ranjan BSC, Lindemann U (2013) A case for multiple views of function in design based on a common definition. AI EDAM 27:271–279.

Chen Y, Liu Z, Huang J, Zhang Z (2013) A multi-agent based framework for multi-disciplinary conceptual design synthesis. In: Proc. Intl. Conf. Engg. Design. Seoul, Korea,

Freedonia Inc. (2015) Demand will be fueled in part by government mandates requiring that all new light vehicles be equipped with electronic stability control and tire pressure monitoring systems. http://www.freedoniagroup.com/DocumentDetails.aspx?ReferrerId=FG-01&studyid=2957. Accessed 19 Jul 2015

George Stalk J (1988) Time—The Next Source of Competitive Advantage. In: Harv. Bus. Rev. https://hbr.org/1988/07/time-the-next-source-of-competitive-advantage. Accessed 10 Dec 2015

Gero JS, Neill TM (1998) An approach to the analysis of design protocols. Des Stud 19:21–61.

Goel A, Bhatta S, Stroulia E (1997) Kritik: An early case-based design system. Issues Appl Case-Based Reason Des 1997:87–132.

Goel AK, McAdams DA, Stone RB (2013) Biologically inspired design: computational methods and tools. Springer Science & Business Media

Goel AK, Rugaber S, Vattam S (2009) Structure, behavior, and function of complex systems: The structure, behavior, and function modeling language. AI EDAM 23:23–35.

Han Y-H, Lee K (2006) A case-based framework for reuse of previous design concepts in conceptual synthesis of mechanisms. Comput Ind 57:305–318.

Helms B, Shea K, Hoisl F (2009) A framework for computational design synthesis based on graph-grammars and function-behavior-structure. In: Proc. ASME IDETC/CIE. pp 841–851
Hirtz J, Stone RB, McAdams DA, et al (2002) A functional basis for engineering design: reconciling and evolving previous efforts. Res Eng Des 13:65–82.
Hsiao S-W, Chen C-H (1997) A semantic and shape grammar based approach for product design. Des Stud 18:275–296.
Ilevbare IM, Probert D, Phaal R (2013) A review of TRIZ, and its benefits and challenges in practice. Technovation 33:30–37.
Jorapur N, Palaparthy VS, Sarik S, et al (2015) A low-power, low-cost soil-moisture sensor using dual-probe heat-pulse technique. Sens Actuators Phys 233:108–117.
Koenigseder C, Stankovi T, Shea K (2015) Improving Generative Grammar Development and Application Through Network Analysis Techniques. Milan,Italy, pp 167–176
Ko WH (1996) The future of sensor and actuator systems. Sens Actuators Phys 56:193–197.
Krishnan G, Kshirsagar CU, Ananthasuresh GK, Bhat N (2012) Micromachined high-resolution accelerometers.
Kurtoglu T, Campbell M, others (2006) A graph grammar based framework for automated concept generation. In: Proc. 9th Intl. Design Conf. Dubrovnik, Croatia,
Kurtoglu T, Swantner A, Campbell MI (2010) Automating the conceptual design process:“From black box to component selection.” Artif Intell Eng Des Anal Manuf 24:49–62.
Latex (2015) LaTeX – A document preparation system. http://www.latex-project.org/. Accessed 24 Jul 2015
Liu Y-C, Chakrabarti A, Bligh T (2000) A Computational Framework for Concept Generation and Exploration in Mechanical Design. In: Gero J (ed) Artificial Intelligence in Design ’00. Springer Netherlands, pp 499–519
Maher ML, Pu P (2014) Issues and applications of case-based reasoning to design. Psychology Press
Malmqvist J, Axelsson R, Johansson M (1996) A comparative analysis of the theory of inventive problem solving and the systematic approach of pahl and beitz. In: Proceedings of the 1996 ASME Design Engineering Technical Conferences.
marketsandmarkets.com (2015) Hall Effect Current Sensor Market by Technology - 2020 | MarketsandMarkets. http://www.marketsandmarkets.com/Market-Reports/hall-effect-current-sensor-market-201606539.html. Accessed 25 May 2016
MathWorks Inc. (2015) Simulink - Simulation and Model-Based Design - MathWorks India. http://in.mathworks.com/products/simulink/. Accessed 23 Jul 2015
Mukerjee A (1991) Qualitative geometric design. In: Proceedings of the first ACM symposium on Solid modeling foundations and CAD/CAM applications. ACM, pp 503–514
Mukerjee A, Joe G (1990) A qualitative model for space. In: Proceedings of the eighth National conference on Artificial intelligence-Volume 1. AAAI Press, pp 721–727
Mukherjee T, Fedder GK (1997) Structured design of microelectromechanical systems. In: Proceedings of the 34th annual Design Automation Conference. ASME, pp 680–685
Nagel JK, Stone RB (2012) A computational approach to biologically inspired design. Artif Intell Eng Des Anal Manuf 26:161–176.
Nagel RL, Vucovich JP, Stone RB, McAdams DA (2007) Signal flow grammar from the functional basis. In: International Conference on Engineering Design, ICED. pp 28–31
Pahl G, Beitz W (1996) Engineering Design - a systematic approach. Springer, New York,
Peysakhov M, Regli WC (2003) Using assembly representations to enable evolutionary design of Lego structures. AI EDAM 17:155–168.
Prabhakar S, Goel AK (1998) Functional modeling for enabling adaptive design of devices for new environments. Artif Intell Eng 12:417–444.
Rihtarsic J, Zavbi R, Duhovnik J (2012) Application of wirk elements for the synthesis of alternative conceptual solutions. Res Eng Des 23:219–234.
Sarkar B (2015) Synthesis of Conceptual Designs. https://github.com/ritz123/SyCd.git. Accessed 30 Aug 2015
Sarkar B, Chakrabarti A, Ananthasuresh GK (2015a) A New Approach to Conceptual Design Synthesis of Sensors. Proc.3rd Intl. Conf. Design Creativity, Indian Institute of Science, Bangalore, pp 192–199
Sarkar B, Chakrabarti A, Ananthasuresh GK (2015b) Synthesis of conceptual designs for sensors using SAPPhIRE-lite. In: Proc. Intl. Conf. Eng. Design. Milan,
Schmidt LC, Shetty H, Chase SC (2000) A graph grammar approach for structure synthesis of mechanisms. J Mech Des 122:371–376.
Schmitt G (1993) Case-based design and creativity. Autom Constr 2:11–19.
Smedley K, Cuk S (1994) Switching flow-graph nonlinear modeling technique. IEEE Trans Power Electron 9:405–413.
Srinivasan V, Chakrabarti A (2009) Sapphire: An Approach to Analysis and Synthesis. In: Proc. Intl. Conf. Engg. Design. Palo Alto, CA, USA, pp 417–428
Srinivasan V, Chakrabarti A (2010) Investigating novelty–outcome relationships in engineering design. AI EDAM 24:161–178.
Starling AC, Shea K (2005) A parallel grammar for simulation-driven mechanical design synthesis. In: Proc. ASME IDETC/CIE. ASME, pp 427–436
Thusu DR (2011) Sensors Market Analysis: Growth Challenges in World Image-Sensors Market. In: Sens. Mark. Anal. Growth Chall. World Image-Sens. Mark. http://www.qualitymag.com/articles/90615-sensors-market-analysis-growth-challenges-in-world-image-sensors-market. Accessed 10 Dec 2015
Transparency Market Research (2015) Wearable Sensors Market: Technological Advancements and Healthcare Concerns to Drive Market at 45.2% CAGR through 2020. http://www.transparencymarketresearch.com/pressrelease/wearable-sensor-market.htm. Accessed 20 Jul 2015
Tung Thanh Bui, Dzung Viet Dao, Nakamura K, et al (2009) Characterization of the piezoresistive effect and temperature coefficient of resistance in single crystalline silicon nanowires. In: Micro-NanoMechatronics and Human Science, 2009. MHS 2009. International Symposium on. pp 462–466
Vattam SS, Helms ME, Goel AK (2008) Compound analogical design: interaction between problem decomposition and analogical transfer in biologically inspired design. In: Design Computing and Cognition’08. Springer, pp 377–396
Voss A, Bartsch-Spörl B, Oxman R, et al (2012) case-based design. Artif Intell Des 172.
Watson I, Perera S (1997) Case-based design: A review and analysis of building design applications. Artif Intell Eng Des Anal Manuf 11:59–87.
Wilson J, Chang P, Yim S, Rosen DW (2009) Developing a bio-inspired design repository using ontologies. In: ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. American Society of Mechanical Engineers, pp 799–808

Wintergreen Research (2014) Semiconductor Wireless Sensor Internet of Things (IoT):Market Shares, Strategies, and Forecasts, Worldwide, 2014 to 2020. http://wintergreenresearch.com/reports/semiconductor%20wireless%20sensor%20networks%202014%20brochure.pdf. Accessed 19 Jul 2015

Wojnarowski J, Kopec J, Zawislak S (2006) Gears and graphs. J Theor Appl Mech 44:139–162.

Wu Z, Campbell MI, Fernández BR (2008) Bond graph based automated modeling for computer-aided design of dynamic systems. J Mech Des 130:041102.

Yan W, Zanni-Merk C, Cavallucci D, Collet P (2014) An ontology-based approach for using physical effects in inventive design. Eng Appl Artif Intell 32:21–36.

Zavbi R (2003) Impact of knowledge twisting on combinatorial explosion. In: Proc. 14th Intl. Conf. Engg. Design. Stockholm,

Zavbi R, Duhovnik J (2000) Conceptual design of technical systems using functions and physical laws. AI EDAM 14:69–83.

Zavbi R, Rihtarsic J (2010) Synthesis of elementary product concepts based on knowledge twisting. Res Eng Des 21:69–85.

Zhou N, Agogino A, Pister KS (2002) Automated design synthesis for micro-electro-mechanical systems (MEMS). In: ASME 2002 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. American Society of Mechanical Engineers, pp 267–273