




Knowledge Representation Framework Combining Case-Based Reasoning with Knowledge Graphs for Product Design

Yingzhong Zhang¹ , Xu Liu², Jia Jia³ and Xiaofang Luo⁴

¹School of Mechanical Engineering, Dalian University of Technology, zhangyz@dlut.edu.cn

²School of Mechanical Engineering, Dalian University of Technology, jhcad@qq.com

³School of Mechanical Engineering, Dalian University of Technology, 49978527@qq.com

⁴School of Mechanical Engineering, Dalian University of Technology, lx@dlut.edu.cn

Corresponding author: Yingzhong Zhang, zhangyz@dlut.edu.cn

Abstract. Product design relies on various design knowledge, including explicit design knowledge and tacit design knowledge. This paper aims at issues that the current design knowledge representation methodology lacks scalability and flexibility, and presents a design knowledge representation framework combining CBR with knowledge graphs. Based on the framework, design cases are represented as a set of knowledge graphs. An ontology model for design case representation is proposed. A novel approach to retrieving design cases from knowledge graphs is presented, which uses a design problem query graph instead of keywords to match similar case subgraphs from knowledge graphs. A semantic similarity assessment method based on subgraph similarity is proposed. Finally a case study on knowledge representation for stamping die design is provided.

Keywords: Design knowledge representation, Case-based reasoning, Knowledge graphs, Semantic retrieval.

DOI: <https://doi.org/10.14733/cadaps.2020.763-782>

1 INTRODUCTION

Product design relies heavily on design knowledge, including various kinds of explicit knowledge and tacit knowledge [3]. The research on knowledge-based intelligent CAD or expert systems has always been attracted. A number of knowledge-based CAD systems have been successfully applied. However, as most design knowledge is implicit, empirical and unstructured, it is very difficult to be captured and formally represented as design rules or design models. As a result, knowledge elicitation has always been considered as the bottleneck of design expert systems [3]. Meanwhile, in general design activities, it is widely accepted that designer's practices rely heavily on their past design experiences, instead of designing everything from scratch [1]. A large number of practices indicate reasoning by reusing past cases is a powerful and frequently applied way to solve new problems for humans [18]. Successful design cases of solutions to design problems in the past should be an important part of design knowledge.

Case based reasoning (CBR) methodology is to solve new problems by adapting previous successful solutions to similar current problems. CBR is similar to human beings' reasoning processes, which can learn over time, and reason with concepts that have not been fully defined or modeled. Aamodt and Plaza [1] described CBR typically as a cyclical process comprising four steps of retrieval, reuse, revising, and retaining. Since CBR methodology was proposed it has been researched intensively. A number of applications [21] have demonstrated CBR is a promising methodology for the knowledge-based product design. Lee and Luo [14] presented a study on die-casting die design based on CBR. Hashemi et al. [10] proposed a case-based reasoning approach for design of machining fixture. Wang and Rong [20] presented a CBR method for welding fixture design. In the CBR method, the retrieval of cases is the most crucial step. In practice, it is not enough to retrieve cases based on a match of keywords, and it needs to understand the really meaning of query problems and to retrieve semantically matched cases with query problems, which requires a conceptualized case representation for different level cases and semantic retrieval that can understand design intent. In addition, sharing and common understanding for cases, and scalability and flexibility of the case base are still issues to be solved.

In order to address above issues in the traditional CBR method, ontology has been introduced into the CBR methodology. Gao and Deng [8] presented a strategy that domain ontology is constructed as the basis of knowledge structures and the concrete case knowledge is represented as the instance of relevant concepts of the domain ontology, through which the knowledge of different systems can be shared. Guo et al. [9] proposed an intelligent retrieval method by integrating ontology technology into CBR systems to design new injection molds. Bejarano et al. [2] proposed an integrated CBR approach based on ontology and preference modeling. Chen et al. [4] proposed an ontology and a CBR based automated decision-making method for the disassembly of mechanical products.

However, along with the continuous deepening of the CBR research and applications, there are the following shortcomings in the current CBR methods for product design:

(1) Lack of high-level conceptual models that can reflect the essence and processes of product design using CBR methodology. Currently, most of CBR research focuses on a specific product design, such as injection mould design and welding fixture design. These CBR systems can only be used to a specific product design.

(2) Lack of semantic hierarchy (or granularity) representation of design cases. In the product design, a design problem is usually decomposed into multiple sub-problems and a sub-problem may continue to be decomposed into multiple its sub-problems, which forms a hierarchical relation tree. A case query may only focus on a terminal leaf of the hierarchical relation tree, or involve in a branch of the tree or whole tree. The current case representation method is unable to provide multi-granularity case retrieval.

(3) Low efficiency and poor flexibility in the design case construction. Current case representation needs a lot of case analysis on design problems and solutions, which is a very tedious work. As a result, designers have no motivation to analyze and summarize the finished design, and knowledge engineers have difficulties in involving tacit design knowledge. In addition, the number of the constructed cases is limited. As a result, it is difficult to satisfy various query requirements for design cases.

Recently, Google presented a knowledge base using knowledge graph (KG) technology to enhance its search engine's results with information gathered from a variety of sources. The objective of knowledge graphs is to create a "web of data" that can be readable by machines [19]. Knowledge graphs model information in the form of entities and relationships between them, and use the W3C Resource Description Framework (RDF) to represent knowledge instances (or facts) in the form of binary relationships. Knowledge graph technology has become a research focus and a large number of knowledge base systems based on knowledge graphs have been created. In addition, ontology can provide a formal and sharing semantic representation for various concepts. The ontology-based knowledge graphs can provide powerful capabilities to formally describe

unstructured facts. Hence, knowledge graphs provide a new feasible solution to formally represent design result facts or cases.

This paper aims at above issues and challenges, focuses on the design knowledge representation of general mechanical products, and presents a design knowledge representation framework combining CBR with knowledge graphs. The main contribution includes using knowledge graphs to represent design cases and presenting a novel design case retrieval method.

The paper is structured as follows. Section 2 gives the overview of the presented knowledge framework. Section 3 presents semantic representation for design cases. Sections 4 presents an approach to retrieving design cases from knowledge graphs. Section 5 provides a case study and discussion.

2 OVERVIEW OF THE FRAMEWORK

According to requirements of design knowledge representation and characteristics of CBR and knowledge graphs, we propose a product design knowledge representation framework combining CBR with knowledge graphs, which is as shown in Figure. 1. This framework consists of a knowledge representation layer and a knowledge operation layer as follows:

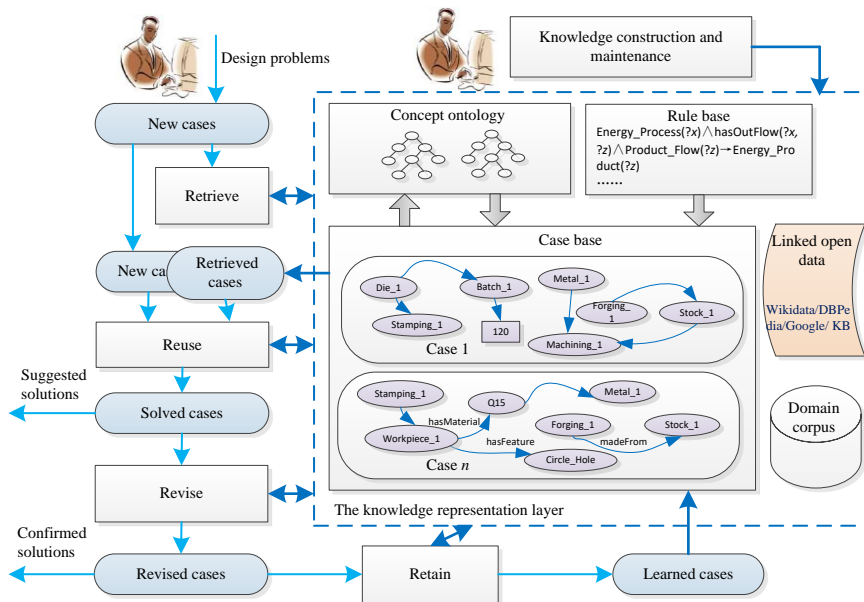


Figure 1: A product design knowledge representation framework.

- Knowledge representation layer.

The knowledge representation layer is the core of the whole framework. It includes three bases: a concept ontology base which provides an explicit description for all concepts and their relations in the knowledge base, a rule base where the causal design knowledge is defined as knowledge rules, and a case base. In the case base, design cases are represented as a set of directed knowledge graphs.

The knowledge representation layer provides a sharable knowledge set with formalized structures for product design. In this framework, the concept ontology base, the rule base and the design case base are described with Ontology Web Language (OWL) and saved in an accessible network space.

- Knowledge operation layer.

The knowledge operation layer is outside the knowledge representation layer, which includes the following two kinds of knowledge operations:

One is the knowledge construction and maintenance operations. In this framework, the domain ontology and the rule base are constructed using Protégé platform [15]. Protégé is a well-known free and open-source ontology editor. Design cases are constructed under the ontological guidance by human-computer interfaces. For this purpose, we developed a knowledge system for stamping die design with Visual C++. The developed knowledge system uses object-oriented programming, and can parse OWL ontology constructed by Protégé and perform semantic query based on backward chained reasoning.

The other is a set of typical CBR application operations, which include retrieving, reuse, revising and retaining of design cases. These CBR application operations have been implemented in our developed knowledge system for stamping die design, which also can be integrated with commercial CAD systems.

This paper mainly focuses on the representation and retrieval of design knowledge combining CBR with KGs.

3 SEMANTIC REPRESENTATION FOR PRODUCT DESIGN KNOWLEDGE

3.1 Ontology-Based Concept Model for Product Design

A concept is an abstract form to represent knowledge entities and their relationships. Knowledge can be described as some interconnected concepts, and each concept is connected through associations. Ontology can provide a formal and sharing semantic representation for various concepts. In the CBR method, a case consists of one knowledge entity or multiple knowledge entities linked by relations. It is necessary to abstract common properties of all cases in a specific design domain and to define them as high-level domain ontologies. With the defined domain ontologies, constructing new design cases can be guided and performed with less ambiguity. In addition, the defined domain ontologies can help case retrieval.

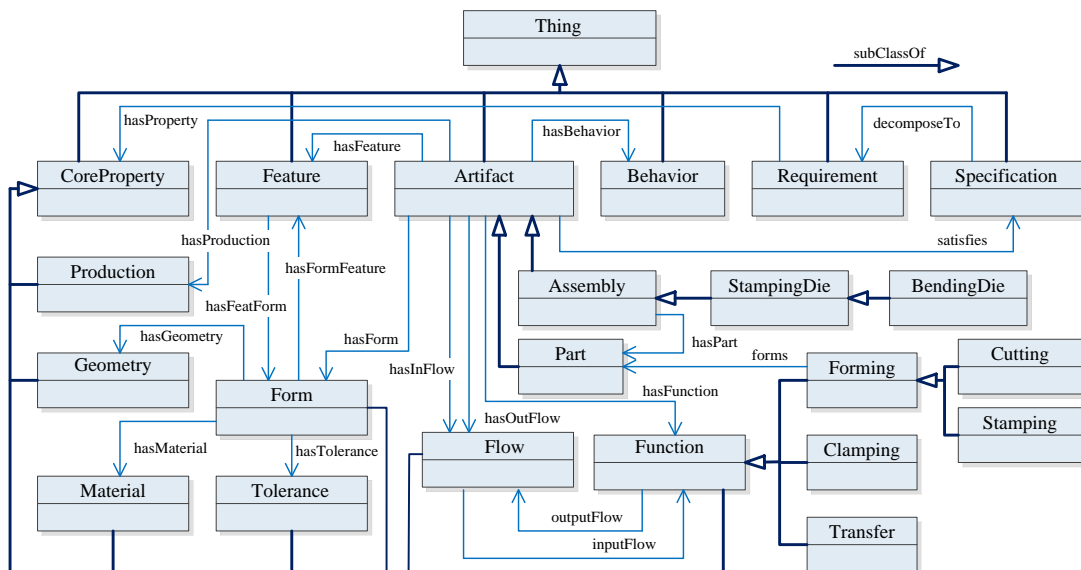


Figure 2: Class diagram of the ontology model for design case representation (portion).

In general, design is a process that constructs a description of an artifact that satisfies functional specifications, meets certain performance criteria and resource limitations. In addition, it is required for a design to be realizable and to satisfy criteria such as simplicity, testability, manufacturability, and reusability. Hence, product design knowledge mainly involves information on design requirements, specifications, functions, forms, and behaviors. Based on literature [7], we defined an ontology-based concept model for case representation as shown in Figure. 2, which includes six top classes: *CoreProperty* class, *Feature* class, *Artifact* class, *Behavior* class, *Requirement* class, and *Specification* class.

3.2 Knowledge Rules for Explicitly Representing Design Knowledge

The ontology defined above explicitly gives the semantic hierarchical relationships for the product design concepts. However, the implied meaning of each concept is not explicitly provided. For example, in the above ontology model, the *StampingDie* class is defined as a subclass of the *Assembly* class, and the *PunchingDie* class and *BendingDie* are defined as the subclass of the *StampingDie* class, which only defines the semantic hierarchical relationships between them. What behaviors or characteristics of a punching die are not explicitly provided. Production rule is an effective method to represent the causal relation knowledge and has been widely applied to various types of knowledge systems. It is very necessary to combine the ontological concepts and instances with production rules. The formalized design knowledge rules can enrich the deductive reasoning capabilities.

The relationships in OWL ontologies can be represented as a set of Description Logic (DL) predicates, which gives the relations precise semantics. In DL, concepts are mapped to unary, relations to binary predicates. The meaning of an element in a relation expression is determined by the domain and range defined in the ontology. In addition, as Semantics Web Rule Language (SWRL) [11] is an expressive OWL-based rule language, in this paper, SWRL is chosen to define the knowledge rules. An SWRL rule contains an antecedent part, which is referred to as the body, and the consequent part, which is referred to as the head. An SWRL reasoning rule can be expressed as a form: Antecedent \rightarrow Consequent, and the antecedent part and consequent part can also be expressed as the conjunctive formula of atoms $a_1 \wedge a_2 \wedge \dots \wedge a_n$ and $b_1 \wedge b_2 \wedge \dots \wedge b_n$, respectively. Atoms a_i and b_i can be either form $C(?x)$ or $P(?x, ?y)$, in which if x is an instance of the class C , then $C(?x)$ holds; If x is related to y by property P , then $P(?x, ?y)$ holds.

In the practical work, the design knowledge is related to some specific applications, which can be represented as knowledge rules encoded with SWRL. These rules give a formal and explicit representation for the implicit design concept knowledge. Table 1 provides a part of knowledge rules on design concepts of stamping dies.

No.	Concepts	SWRL rules	Explanation
1	PiercingDie	$\text{StampingDie}(?x) \wedge \text{hasFunction}(?x, ?f) \wedge \text{Punching}(?f) \wedge \text{forms}(?f, ?p) \wedge \text{Workpiece}(?p) \wedge \text{outputFlow}(?f, ?u) \wedge \text{differFrom}(?u, ?p) \rightarrow \text{PiercingDie}(?x)$	The punched out piece is separate from the output workpiece.
2	BlankingDie	$\text{StampingDie}(?x) \wedge \text{hasFunction}(?x, ?f) \wedge \text{Punching}(?f) \wedge \text{forms}(?f, ?p) \wedge \text{Workpiece}(?p) \wedge \text{outputFlow}(?f, ?u) \wedge \text{sameAs}(?u, ?p) \rightarrow \text{BlankingDie}(?x)$	The blanked out piece is same as the output workpiece.
3	BendingDie	$\text{StampingDie}(?x) \wedge \text{hasFunction}(?x, ?f) \wedge \text{Blending}(?f) \wedge \text{forms}(?f, ?p) \wedge \text{Workpiece}(?p) \wedge \text{outputFlow}(?f, ?u) \wedge \text{sameAs}(?u, ?p) \rightarrow \text{BendingDie}(?x)$	The blended workpiece is same as the output workpiece.

Table 1: Partial knowledge rules encoded with SWRL for stamping die design.

3.3 Semantic Representation for Design Cases Using Knowledge Graphs

Case representation is the foundation of a CBR system as it directly affects the retrieval, reuse, and retain of cases. According to traditional CBR terminology, a case usually denotes a problem situation [1]. In the most current research, the CBR problem situation is usually represented as a set of problem–solution tuples or problem–solution–evaluation triples [5, 12]. However, there are the following issues: (i) a complex design problem usually needs to be decomposed into a hierarchy of sub-problems, which requires a formal structure to represent the problems and the relations between them; (ii) it is difficult or not an easy task for a designer to extract and define the design problems and the associated solutions, which results in poor efficiency of design case construction; and (iii) in the design process, many design requirements and problems are not well identified, so it is difficult to carry out comparisons with feature-value pairs of the problem–solution.

In the light of above issues, we propose an approach to represent design cases using knowledge graphs. The following gives a definition on the design case.

Definition 1. A design case is a set of factual instances interrelated by design results and relations between them on a topic within a product design. The topic determines the level and granularity of a design case. A design case is represented as a design knowledge graph or a segment of the graph.

A design case is represented by a graph $G = \langle V, E \rangle$, where V is a set of design instance nodes and E is a set of directed relation edges that link two instances. The design instances are usually labeled by ontologies or resources. The design case base consists of a set of independent graphs: $CB = \{G_1, G_2, \dots, G_n\}$.

Using knowledge graphs to represent design cases has the following characteristics:

- Design cases are classified as explicit design cases and implicit design cases.

An explicit design case is a design KG defined in the case base, which is explicitly constructed by designers according to design results and design domain ontology. Implicit design cases implicitly exist in a design KG, which can be retrieved according to different design queries. Obviously, an implicit design case is a subgraph of an explicit design KG, which may be a triple or a tree branch of a graph. For example, a KG on stamping die design includes many design cases like material selection of a convex-die part.

- The design case graph can represent the hierarchy of design cases.

In general, cases can be represented using multiple representations at different levels of abstraction. The basic idea behind these approaches is to represent a case at multiple levels of detail, possibly using multiple vocabularies. When a new problem must be solved, similar cases at appropriate levels of abstraction are retrieved from the case base, and solutions from these cases can be obtained.

With the knowledge graph structure, various design cases with different semantic granularity can be easily extracted.

- The design case construction only needs to record the final design results of a product under the ontological guidance.

It is easier and more straightforward to construct design cases from recording the design results than from extracting design problems and solutions. The problem and solution are implicitly embedded in the semantics of relations between instances. As a result, the CBR system is much more flexible and scalable. The design case base is much easier to be constructed.

4 RETRIEVING DESIGN CASES FROM DESIGN KNOWLEDGE GRAPHS

4.1 Overview for the Retrieval Method

In the CBR cycle, an important step is to retrieve previous cases that can be used to solve the new target problem. In the knowledge representation framework presented in this paper, the design

cases are represented as a set of KGs and the design problems are not predefined, which can represent and retrieve design cases with much more flexibility and scalability. However, it is required to employ a method that is different from past retrieval method. In this paper, we propose a new method to retrieve design cases as shown in Figure. 3. The presented method mainly includes the following procedures:

(1) Design problem description.

In general, a complex design problem involves multiple design requirements and constraints. Currently, it is very difficult to use keywords to describe a complex design problem. Hence, at first we need to decompose a design problem into multiple simple design problems consisting of one concept or one relation, which can form a problem query graph.

This is a very crucial step, which can be implemented by human-computer interaction systems via the template pattern. Then, the problem query graph is converted into a design query statement represented by first-order logic.

(2) Matching the design problem graph with design case KGs.

In this procedure, the constructed problem query graph is used to match design case KGs in order to find similar design cases. The match is a process to search an isomorphic subgraph to the problem query graph from KGs. If the match is successful, the whole graph where the matched sub-graph locates is the desired design case. However, for a new design, there is no such a subgraph that is completely consistent with the problem query graph (including structures and contents) in previous case graphs. We can only obtain a set of similar design cases.

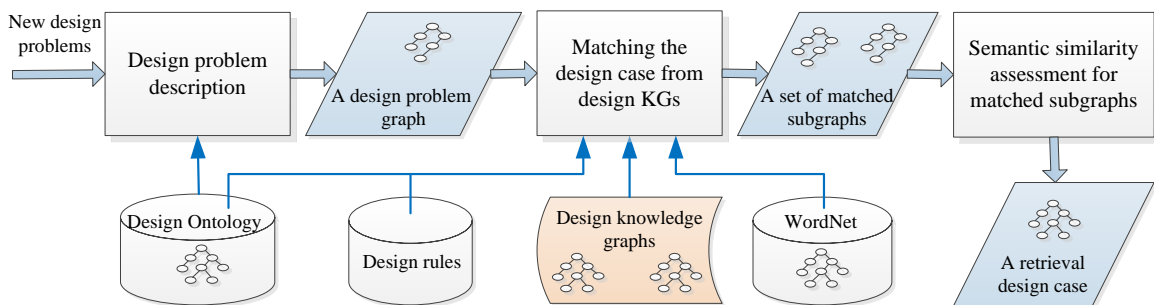


Figure 3: Overview of retrieving a design case from KGs.

(3) Semantic similarity assessment.

In this procedure, the design cases with low similarity obtained in above steps need to be further carried out semantic similarity assessment in order to retrieve the most similar design case from the case candidates. The semantic similarity assessment is implemented between the problem query graph and isomorphic subgraphs in the case candidate.

Semantic similarity assessment between two graphs is relatively difficult. We propose an approach to ranking subgraphs by combing node similarity and node weight, which can obtain the most similar design case. The following will detail the above three procedures.

4.2 Design Problem Representation

In the product design, the initial design problem usually comes from design requirements. In order to meet design requirements or design tasks, a set of design solutions are required to be found. On the other hand, design requirements usually come from design tasks and product markets, which include function requirements, performance requirements, structure requirements and sustainability requirements. Naturally, the initial design problem needs to describe these design

requirement information. In addition, design constraint, manufacturing, and environment information are also needed to be considered.

In general, a design problem is described by natural language. For example, a design problem is described with natural language as: "Design a stamping die that can punch a group of circular holes on a steel sheet; the hole diameter is 20 mm, the thickness of the sheet is 3 mm, the sheet material is Q235, and the production batch is 10,000 pieces". Obviously, it is difficult to formally describe this kind of design problems, which involves multiple design variables and design parameters. Traditional CBR methods usually predefine the problems in the design case with a specific pattern and use the index as retrieval keywords. As a result, this kind of CBR methods lacks scalability and flexibility.

In fact, a new design problem can be regarded as a set of targeted design facts. We can describe the design problem facts as a set of RDF triples, which is same as design cases in design KGs. For example, the design problem sentence mentioned above can be represented as a graph consisting of multiple triples as shown in Figure. 4. It can be seen that the RDF graph pattern can provide a formal and unified description for unstructured design problems.

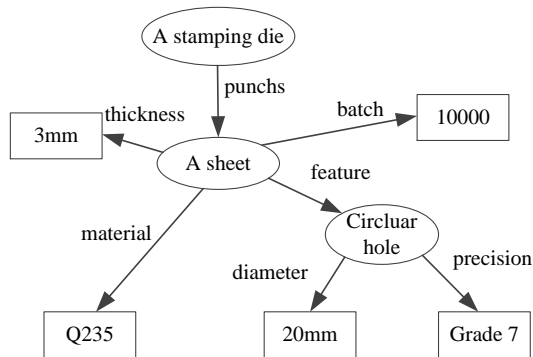


Figure 4: RDF triple graph representation for a design problem.

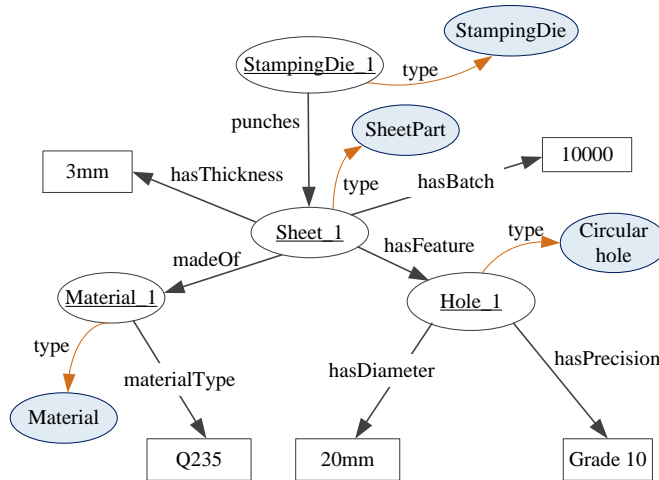


Figure 5: A design problem RDF graph labeled with ontology.

Furthermore, we can label the entities and data as instances of ontological concepts as shown in Figure. 5. It can be seen that "A stamping die" is labeled as an instance of the *Assembly* class, and the thickness value is labeled as attributes value of the instance *Sheet_1* which is type of the *SheetPart* class, which can help realize the variable representation for design parameters. The labeled design problem RDF graph has much more semantic information, which can help the subsequent semantic retrieval for design cases.

4.3 Matching the Design Problem Graph from Knowledge Graphs

In the CBR methodology, the most crucial step is the case retrieval. According new design problems previous design cases are retrieved. In traditional CBR methods, the design cases are usually indexed according to some design attributes, and the case retrieval is implemented according to the similarity between case features and keywords. In this paper, we use design problem RDF graph instead of keywords to retrieve design cases from design KGs, which is a complete semantic relational retrieval. Compared with the traditional CBR method, the presented method has much more flexibilities, scalabilities, and capabilities to perform knowledge reasoning.

First of all, we convert the retrieval of design cases into an issue on semantic query from design KGs. The semantic query language for RDF is known as SPARQL [22]. SPARQL allows users to compose structured queries consisting of triple patterns, where a triple pattern is an RDF triple with one or more variables. A variable occurring in one of the triple patterns can be used again in another triple pattern in the query, denoting a join condition. For example, above example's information need is to find a stamping die design case that can meet the new design requirements. The above example could be formulated by the following SPARQL query:

```
SELECT ?s WHERE {
```

```
  ?s type StampingDie. ?s punches ?w. ?w type SheetPart. ?w hasBatch 10000. ?w hasThickness 3. ?w madeOf ?m. ?m materialType Q235. ?w hasFeature ?f. ?f type CircularHole. ?f hasDiameter 20. ?f hasPrecision 7}
```

where *?s*, *?w*, *?m*, *?f* are variables, and eleven triples are used in the subject-predicate-object (SPO) pattern. The query results are all the subgraphs of the underlying knowledge graph that are isomorphic to the query graph.

However, using above SPARQL pattern to query design case from design KGs, the query results may be disappointment. There are two reasons: One reason is that in the previous design case it is unlikely that design parameters are exactly same to new design requirements. In the engineering design, the design parameters usually vary in an interval. For example, in a specific product design, the value of thickness of a sheet is usually limited in the interval from 0.1mm to 10mm. Hence, the design parameter that is located in the design interval should be regarded as meeting the query requirements. What design interval is defined involves in design knowledge. The other reason is that the SPARQL language supports expressive and precise queries over RDF graphs of entity relationship. As a result, its results are affected by the matching between properties and facts. In addition, SPARQL lacks reasoning capabilities combining knowledge rules.

On the other hand, Semantic Query-enhanced Web Rule Language (SQWRL) [16], a SWRL-based query language, has been widely used. SQWRL takes a standard SWRL rule antecedent and effectively treats it as a pattern specification for a query. An implementation of SQWRL has been developed in the SWRLTab plugin in Protégé-OWL. Based on referring SQWRL and under the Closed-World Assumption, we developed a semantic query approach like SQWRL using backward chained reasoning [25]. According to the SQWRL pattern, above query example can be represented as the following form:

```
StampingDie(?s) ^ punches(?s, ?w) ^ SheetPart(?w) ^ QBatch(?w) ^ madeOf(?w, ?m) ^
QMaterial(?m) ^ QThickness(?w) ^ hasFeature(?w, ?f) ^ CircularHole(?f) ^
QDiameter(?f) ^ QPrecision(?f) → select(?s)
```

It can be seen that a SQWRL statement can be divided into two parts. The right hand side is a core SQWRL operator: "select". It takes one or more arguments, which are typically variables used in the pattern specification of the query, and builds a table using the arguments as the columns of the table. The left hand side likes a standard SWRL rule antecedent with its associated semantics. Each atom will match not only all OWL individuals that are directly of its class but will also match individuals that are entailed by the ontology to be individuals of that class. In fact, all variables that would be bound in a SWRL rules antecedent will also be bound in a SQWRL pattern specification.

After execute above query statement, the select operator will return a list of instances of stamping dies, which can be selected as a set of design case candidates.

In the above query statement, *QBatch*, *QMaterial*, *QThickness*, *QDiameter*, and *QPrecision* are derived concept classes. Specifically, these concept classes are not defined in the ontology base, but defined in the design rule base. These derived concepts can be defined using SWRL according to specific design knowledge, and be used to query in the SQWRL query statement. Table 2 lists the five derived concepts.

No.	Concepts	SWRL rules	Explanation
1	QBatch	SheetPart(?x)∧hasBatch(?x, ?y)∧greaterThan(?y, 1000)∧lessThan(?y, ?100000) → QBatch(?x)	The lot size is greater than 1000 and less than 100000.
2	QMaterial	Material(?x)∧materialType(?x, ?m)∧enumerate(?m, ?tb) → QMaterial(?x)	The material is a commonly used material.
3	QThickness	SheetPart(?x)∧hasThickness(?x, ?y)∧lessThan(?y, ?20) ∧greaterThan(?y, 0.1) → QThickness(?x)	The thickness is greater than 0.1mm and less than 20mm.
4	QDiameter	CircularHole (?x) ∧hasDiameter(?x, ?y) ∧ greaterThan(?y, 0.1)∧lessThan(?y, ?200) → QDiameter(?x)	The diameter is greater than 0.1mm and less than 200mm
5	QPrecision	Feature(?x)∧hasPrecision(?x, ?y)∧lessThan(?y, ?13)∧greaterThan(?y, 2) → QPrecision(?x)	The precision is greater than grade 2 and less than grade 13

Table 2: Knowledge rules encoded with SWRL for five derived concepts.

4.4 Similarity Assessment for Design Cases

By executing the semantic query operation provided in above Section, we can obtain a set of design case candidates which includes design solutions. The next step is to find a most similar design case to designer's desires. From engineering practices, the similarity measure of design cases mainly involves two aspects: semantic similarity measurement and numerical similarity measurement.

Semantic similarity is a concept whereby a set of documents or terms within term lists are assigned a metric based on the likeness of their meaning or semantic content [23]. As the importance of semantic similarity, a large number of methods on semantic similarity measures have been proposed, such as node distance, information content, and hybrid methods [23]. As in the above method both the query problem and the retrieved candidate design cases are labeled by same domain taxonomy ontology, node distance metrics can largely obtain the semantic similarity between two concepts associated by instances.

On the other hand, the semantic similarity of concepts is a qualitative comparison at abstract level. However, in the engineering design, due to the difference of numerical values of some key attributes, design scheme may be completely different. Hence, numerical similarity measurement is also very important for obtaining the most similar case.

Based on above analysis, we perform similarity assessment for design cases according to the following strategies:

4.4.1 Semantic similarity measurement for concepts

First, we calculate the semantic similarity of the concept pair (c_i, c_j) where $c_i, c_j \in C$ (a domain ontology concept set), which is formally defined as: $sim(c_i, c_j)$. The most intuitive semantic information is the semantic distance between concepts, which is usually represented by the path connecting two concepts in the domain ontology.

Let Path (c_i, c_j) be the set of paths connecting the concept c_i and c_j . Let $|P_i|$ denote the length of the path, then $length(c_i, c_j) = \min(|P_i|)$ denotes the shortest path length between two concepts. Using the path based method, the semantic similarity of the concept pair (c_i, c_j) is represented as

$$sim(c_i, c_j) = \frac{1}{1 + length(c_i, c_j)} \quad (4.1)$$

If the concept c_i and c_j have a common parent node, $length(c_i, c_j)$ return 0, and their semantic similarity is 1. Hence, if the concepts are at the same level and no path exists, their distance is 0. For example, if the material of a sheet is "Q235" and the material of another sheet is "Q235A", due to the fact that "Q235" and "Q235A" are type of metal materials, their concept distance is 1, which indicates their semantic is similar.

In general, in the design case instance, if the instance is a string data, and the meaning of the string data needs to be interpreted by the related concept, it is need to be compared from the concept level.

4.4.2 Numerical similarity measurement for attributes

In general, the numerical values of concept attributes are mainly classified into three types: certainty numerical values, interval values and fuzzy numerical values [9].

- Numerical similarity measurement for certainty values.

If n_i is a certainty value of a numerical type attribute from the target case and n_j the one from the source case, their numerical similarity is:

$$sim_{CN}(n_i, n_j) = 1 - \frac{|n_i - n_j|}{k \max(n_i, n_j)} \quad (4.2)$$

where $sim_{CN}(n_i, n_j)$ represents the numerical similarity for certainty values; k is the value scale of the attribute.

- Numerical similarity for interval values.

If the interval $[a_1, a_2]$ is an attribute value of the target case and $[b_1, b_2]$, the one of the source case, the numerical similarity for the interval value is:

$$sim_{IN}([a_1, a_2], [b_1, b_2]) = 1 - \left\{ \frac{1}{2} \left[(b_2 - a_2)^2 + (b_1 - a_1)^2 \right]^{\frac{1}{2}} \right\} \quad (4.3)$$

where $sim_{IN}(\cdot, \cdot)$ is the similarity of the interval type attribute, $a_1 < a_2$ and $b_1 < b_2$.

- Numerical similarity for fuzzy values.

In engineering design, some attributes may use a fuzzy numerical value. In general, the fuzzy value includes the target value v and some relations. For example, a value is represent as ">30", in which the target value is 30 and the relation is ">". It is necessary to measure their fuzzy numerical similarity. In this paper, we employ Triangular function (TriF) [17] to measure the fuzzy numerical similarity.

Let f be a feature (an attribute), $dom(f)$ is the domain scope of this feature in the case-base, $\min(f)$ and $\max(f)$ represent, respectively, the lowest and uppermost domain limits. The fuzzy numerical similarity is calculated with the following formula as

$$sim_{FN}(z) = \begin{cases} \frac{z - \min(f)}{v - \min(f)} & \forall z \in [\min(f), v) \\ 1 & z = v \\ \frac{\max(f) - z}{\max(f) - v} & \forall z \in (v, \max(f)] \\ 0 & \text{others} \end{cases} \quad (4.4)$$

where v is a target value of the fuzzy numerical value for a feature, z is a real value for the same feature of source case, and $sim_{FN}(z)$ represents the similarity between them.

4.4.3 Ranking matched subgraphs based on similarity assessment

As mentioned above, by matching the design problem query graph from KGs, a set of matched subgraphs are retrieved from design KGs. Each subgraph is related to a design case. We only need to find the most similar matched subgraph, because the whole graph where the matched subgraph locates is the design case to be retrieved. Hence, the issue on the similarity measure between the design problem and the design case candidate becomes a similarity measure between two directed relational graphs. In addition, due to the retrieval pattern used by this approach, the graph structure and the node type of the two matched graphs are exactly same.

According to SimRank similarity theory [13], "two objects are similar if they are related to similar objects". It can be concluded that the node and relation in the two graphs are similar, the two graphs are similar. Obviously, the similarity of two graphs relies on the similarity of their nodes and their graph structures. Hence, we can calculate the similarity of nodes in the two graphs to be compared according to above methods respectively. The computed similarity results also form a hierarchical graph same as the matched subgraph, which is shown in Figure 6.

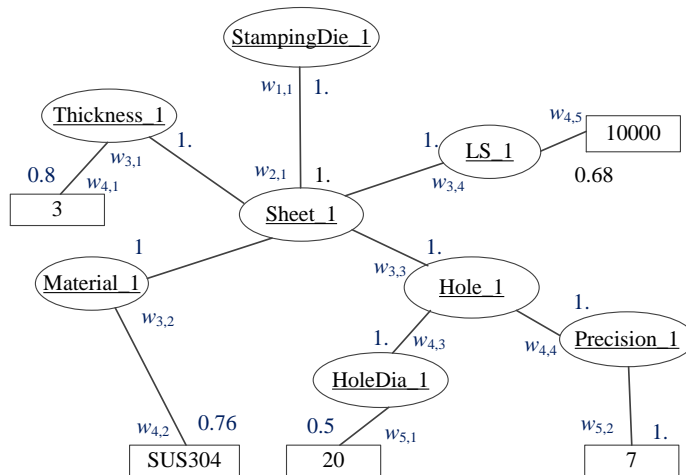


Figure 6: Illustration for similarity distribution on the matched subgraph.

In general, the similarity of nodes located on different levels will have different contributions to the similarity of the whole graph. Specifically, the similarity contribution of nodes on the top level is greater than on the bottom level [26]. In addition, the different nodes may have different connecting edges; obviously, the node that has more out connecting edges (or out-degrees) has much more influences on the similarity to the compared graph. Hence, we introduce the node weight to represent the similarity influence for each node.

Define 2: The number of edges from a node to its tree's root node is defined as the depth of the node.

Define 3: The depth of a node plus 1 is defined as the level of the node.

Let m be the number of the level of a tree. The weight of each node can be calculated according to the following formula:

$$w_{i,j} = \begin{cases} \frac{m-i+1}{m} & E_{i,j} = S_i \\ \left(1 + \frac{E_{i,j}-1}{S_i}\right) \frac{m-i+1}{m} & E_{i,j} \neq S_i \end{cases} \quad (4.5)$$

In formula (4.5), $i=1,2,\dots,m$, $w_{i,j}$ denotes the weight of the j th node located on i th level, $E_{i,j}$ denotes the number of edges starting from the node (or the out-degree of the node), and S_i denotes the total out-degree of all nodes on the i th level. For example, the weight of node "Material_1" in Figure 6 can be calculated as follows:

$$w_{3,2} = \left(1 + \frac{1-1}{5}\right) \left(\frac{5-3+1}{5}\right) = \frac{3}{5} = 0.6$$

The weight of the root node is 1. As a result, the similarity between query problem graph and k th matched subgraph is calculated from the following formula:

$$\text{sim}(G_q, G_k) = \sum_{i=1}^m \sum_{j=1}^{N_{i,j}^k} w_{i,j} \text{sim}(N_{i,j}^k) \quad (4.6)$$

In formula (4.6), $N_{i,j}^k$ denotes the similarity of the j node on the i level, and $w_{i,j}$ denotes the weight of the node.

All matched subgraphs and the query problem graph are performed above similarity computation. The similarity value is ranked, and the subgraph with maximum similarity value is considered as the most similar subgraph. The knowledge graph where the most similar subgraph locates is retrieved.

5 CASE STUDY AND DISCUSSION

5.1 Stamping Die Design Case Representation and Retrieval

In order to better illustrate the presented approach, we take the design case representation of stamping dies as the case study. A stamping die is a kind of widely used technological equipment that can process sheet metals into workpieces or semi-finished products. Its design heavily depends on the shape, material, accuracy and lot size of the workpiece to be formed. According to the representation framework provided above, the procedures on design case representation and retrieval of stamping dies are as follows:

(1) Constructing domain ontology of stamping die design.

This knowledge representation framework provides general and fundamental design ontology. The stamping die ontology can be built on this platform as shown in Figure 7 (in order to explain problems and simplicity, only a few of necessary concept classes are listed and some properties are omitted).

From Figure 7, we can see that a stamping die class is defined as a sub-class of the *Assembly* class. By means of properties (or relations) a design instance of stamping dies is related to its workpieces (by "forms" relation) and its structures (by "hasPart"). The stamping die ontology provides a conceptual semantic description for design factual instances, and guidance for subsequent construction of design cases.

(2) Constructing design case KGs under the guidance of the defined ontology.

According to the classes and properties defined in ontology, designers are guided to construct design factual instances step by step, which is only a labeled recorder of design results. From Figure 7, a design instance of the *StampingDie* class constitutes a stamping die design case, which can completely describe design result facts on this product. In this experiment, we constructed design cases of three stamping dies, which are two punching dies and one blending die. The structure information of stamping dies are represented as instances of the *Assembly* class and *Part* class, respectively, which are related to the product design instance by the "hasPart" property (or relation). In general, structural part of stamping dies includes convex-die, concave-die, die carrier, die holders, positioning parts, guiding parts, etc. The following lists a part of design instances:

ABox = {PunchingDie{*S1*, *S2*}, BlendingDie{*S3*}, SheetPart{*W1*, *W2*, *W3*}, hasBatch{(*W1*, 1000), (*W2*, 5000), (*W3*, 8000)}, forms{(*S1*, *W1*), (*S2*, *W2*), (*S3*, *W3*)}, CircularHole{*F1*, *F2*}, Bending{*F3*}, hasFeature{(*W1*, *F1*), (*W2*, *F2*), (*W3*, *F3*)}, hasDiameter{(*F1*, 10), (*F2*, 15)}, hasRadius{(*F3*, 5)}, hasPrecision{(*F1*, 9), (*F2*, 12), (*F3*, 13)}, Steel{*M1*, *M3*}, Aluminum{*M2*}, madeOf{(*W1*, *M1*), (*W2*, *M2*), (*W3*, *M3*)}, materialType{(*M1*, Q235), (*M2*, 1035), (*M3*, SGCC)}, hasThickness{(*W1*, 2), (*W2*, 3), (*W3*, 1.5)}, hasPart{(*S2*, *C2*), (*S2*, *H2*), (*S3*, *C3*), (*S3*, *H3*)}

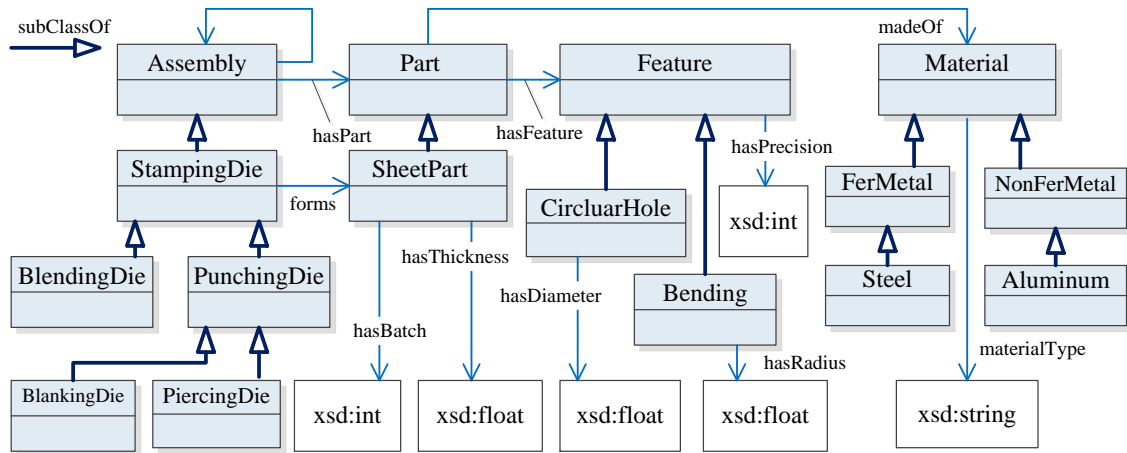


Figure 7: A part of stamping die classes and their properties.

We can represent the facts in above ABox as three RDF graphs as shown in Figure 8, which constitute three design case graphs.

(3) Retrieving design cases from design knowledge graphs.

If a new problem is to design a stamping die and demands that the die can pierce a set of circular holes on a sheet workpiece, the batch of the production is 1600, the thickness of the sheet is 2.5mm, the diameter of the circular hole to be punched is 10mm, the requirement for precision is grade 10, and the sheet material uses Q235A, we can retrieve previous design cases from above KGs.

First, we retrieve all stamping dies that can forms circular hole features on a sheet part, which can be represented as the following query statement:

StampingDie(?s) ^ forms(?s, ?w) ^ SheetPart(?w) ^ hasFeature(?w, ?f) ^ CircularHole(?f) → select(?s)

Above query can be implemented using a backward chained reasoning. In fact, from the facts in the above ABox, the instance of *StampingDie* class that can meet left condition requirements of above query statement is instance *S1* and *S2*. As a result, from above query we can obtain two sets of stamping die instances. The following step will rank the most similar stamping die according to their design parameters.

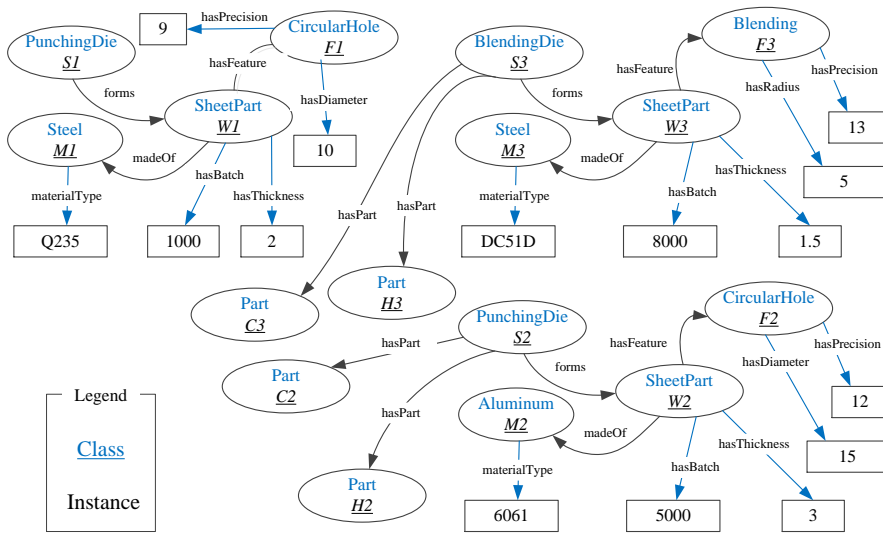


Figure 8: Three design case knowledge graphs for stamping dies (a part of graphs).

(4) Ranking retrieved design candidates

After the die instance has been retrieved, all its related information can be easily obtained by further queries. Using the similarity assessment method provided above Section, the following steps are implemented:

- Constructing the design problem query graph.

All design requirements are represented as a design problem query graph, which is denoted by g_q and as shown in Figure 9. At the same time, according to relations in g_q , we can easily obtain two design attribute relation graphs of the retrieved die instance $S1$ and $S2$, which are also called as matched subgraphs and are denoted by g_1 and g_2 . The structure of the matched subgraph is same as the query graph.

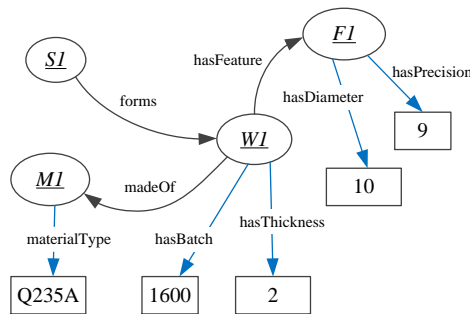


Figure 9: The design problem graph.

- Computing the similarity of each node between the query graph and the matched subgraphs.

Using Equation (4.1) and Equation (4.2) provided above Section, compute the node similarity between g_q and g_1 , and between g_q and g_2 , respectively.

- Ranking design cases candidates.

Using Equation (4.5) provided above Section, compute the weight of each node. We can obtain two node similarity graphs as shown in Figure 10. In Figure 10, we replace the node content of original graphs with node similarity, and the numerical data next to a node is the weight value of the node.

According to Equation (4.6), the calculated similarity value is as follows:

$$\text{sim}(g_q, g_1) = 7.765$$

$$\text{sim}(g_q, g_2) = 6.217$$

Above calculated results show that g_1 is the most similar subgraph to the query graph. As a result, the related stamping die instance $S1$ is the retrieved case. The calculated ranking result is consistent with expectations.

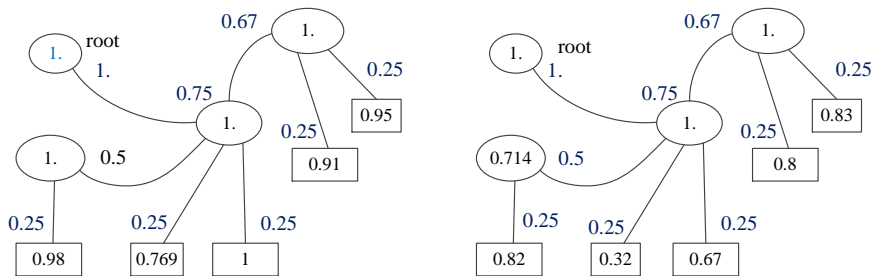


Figure 10: Node similarity graphs: (a) The node similarity graph between g_q and g_1 , (b) The node similarity graph between g_q and g_2 .

5.2 Comparison and Discussion

In order to demonstrate the advantages of the presented approach we select the following two examples as comparison targets, which use traditional CBR methods:

Literature [6] uses typical traditional CBR methods to aid special purpose machine (SPM) design. In the CBR method, the design case base is divided into workpiece case base and the SPM case base. The former includes previous cases of workpieces and the latter the previous solutions of SPMs for these cases that can be reused in a new design case. Workpiece attributes and machining attributes of a design case are indexed into a 13-bit code such as "2122423132212". Each number in different bits refers to a specific meaning; for example, "2" in the first bit represents the workpiece is a cubic component and "1" in the second bit represents the workpiece shape is a plane.

Similarly, literature [24] uses typical CBR methods to aid body-in-white fixture design. A fixture case is also indexed into a 15 bit code such as "CB1122131115142. Each character in different bits refers to a specific meaning; for example, the number in third bit represents the type of drive power, where 1 is non-driven, 2 is pneumatic, 3 is manual toggle, 4 is pneumatic toggle, and 5 is other.

Obviously, the above indexing of design cases is a hard coding pattern. First, the indexed code bits are fixed, which are not easily changed. As a result, it is very difficult to add or remove attribute information of the defined design cases. Second, each character in different bits of a case code refers to a specific meaning. The subsequent retrieving programs have to obey these specific meanings. Furthermore, a one-dimensional string code lacks hierarchical and granulometric representation for the design attribute information.

If the approach presented in this paper is employed, according to the meaning of each code provided by literature [6] the case indexing code "2122423132212" can be represented as a KG as shown in Figure 11. It can easily be seen that the presented approach is much more flexible and

scalable. Design attributes can be easily added or removed, which also not affects the subsequent retrieval of design cases.

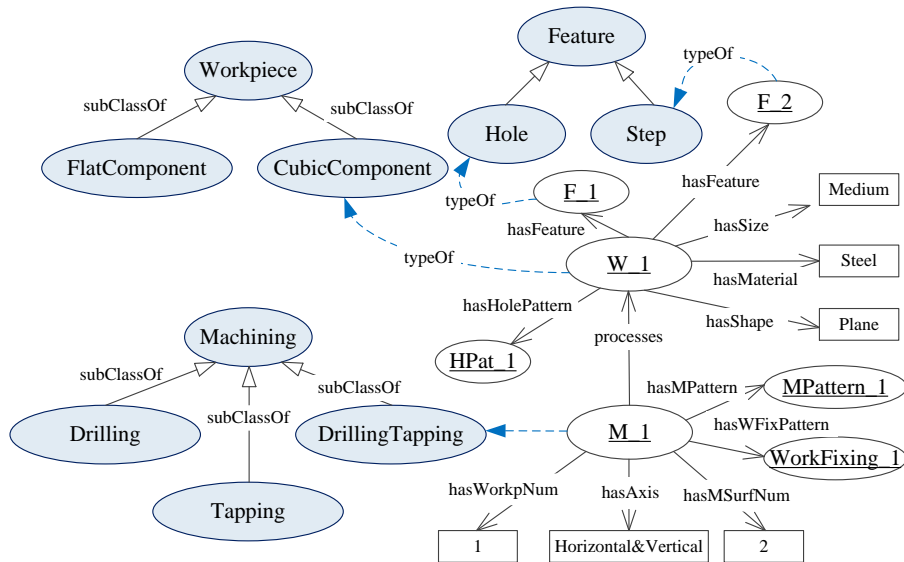


Figure 11: KG representation for the design case encoded with “2122423132212” [6].

From above approach analysis and comparison, the presented approach has the following advantages.

- Design KG representation can provide an ontology based formal solution to represent empirical and unstructured design knowledge.

It is well known that it is very difficult to represent empirical and unstructured design knowledge. Traditional CBR methods need to index case information into a code according to predefined rules. Retrieval of cases depends on the indexing code that is difficult for human and computers to understand. In the presented method, whether the design query or design result facts are represented in RDF triple pattern, which doesn't need a predefined and fixed design case code, and is close to the natural language expression form that is familiar to human beings. In the further work, we will realize the automatic transformation from the problem query statement in natural language to problem query graphs.

In addition, domain ontology provides a conceptual specification with much closer to product design practice. Concept is the fundamental knowledge unit. The transfer and exchange of knowledge is mainly realized by understanding of concepts. Hence, various concepts need a formal and sharing semantic interpretation. Currently, ontology can satisfy this requirement. However, due to different applications, the understanding for same a concept is different. Hence, it is necessary to construct domain ontology for specific product design. For example, in the above case study, “Q235” and “6061” denote two types of metal materials, but there are some differences between them, which is required to define their related concepts on the different ontological level.

- The presented approach is much more flexible and scalable than traditional CBR methods.

From above case comparison, we can see that in the traditional CBR methods, indexing code of design cases is defined according to predefined rules. The bit number of an indexing code and the meaning of each bit are predefined and difficult to change. In practice, with the development of technology, design case information is also constantly changing. In the presented approach, design attribute information can be easily added or removed only by adding or removing an edge in the design case KGs.

- The presented approach has semantic hierarchy (or granularity) representation abilities for design cases.

In the product design, a design problem is usually decomposed into multiple sub-problems and a sub-problem may continue to be decomposed into multiple sub-problems, which forms a hierarchical relation tree. A case query may only focus on a terminal leaf of the hierarchical relation tree, or involve in a branch of the tree or whole tree. The current CBR case representation method is unable to provide multi-granularity case retrieval.

Except above advantages, the presented approach also has the following issues that need to be addressed:

- The matching and semantic similarity assessment between complex subgraphs need much more intelligent technology supports.

The efficient and accurate matching and semantic similarity assessment between complex subgraphs are still issues to be addressed. It is not enough only to consider the structural distribution of the node in the subgraph to determine its similarity weights. In some cases, the node with less out-degree may have more effects than those with more out-degrees on some topic's similarity.

- The presented KGs based CBR method still belongs to symbol-based reasoning.

Currently, it is still difficult to represent shape information of design cases using the symbol-based reasoning. Combining the presented approach with data-driven deep learning technology is a new challenge.

6 CONCLUSION

CBR methodology provides a feasible solution for implicit, empirical and unstructured design knowledge to be captured and formally represented. However, the traditional CBR method lacks scalability and flexibility, and the efficiency to define and construct design cases is low as it is need to carry out a large number of analysis works on design problems and associated design solutions according a specific pattern. This paper aims at these issues and presents a design knowledge representation framework combining CBR with knowledge graphs. Based on the framework, design cases are represented as a set of design knowledge graphs based on ontology, which only records the design result facts under the guidance of the defined ontology model. An ontology model for design case representation is proposed. A novel approach to retrieving design cases from knowledge graphs is presented. In the presented approach, the design problem is represented as a RDF graph instead of retrieval keywords; the design case retrieval is converted a semantic relational query issue. The query results are all the subgraphs of the underlying knowledge graph that are isomorphic to the problem query graph. By means of semantic similarity assessment for the query results, a most similarity design case can be retrieved. Finally a case study on knowledge representation for stamping die design is provided, which shows the presented method is feasible.

The presented knowledge representation framework provides a new solution to represent empirical and unstructured design knowledge. In the future work, we will focus research on the understanding and conversion of a design problem query represented in natural language. In addition, the design case retrieval based on fuzzy similarity and knowledge reasoning is also needed to do in-depth research.

ACKNOWLEDGEMENTS

This work is supported by the National Science Foundation of China (Grant No. 51775081). The authors thank the anonymous reviewers for their helpful suggestions on this study.

Yingzhong Zhang, <http://orcid.org/0000-0003-3584-7239>

REFERENCES

- [1] Aamodt, A.; Plaza, E.: Case based reasoning: foundational issues, methodological variations, and system approaches, *AI Communications*, 7(1), 1994, 39-59.
- [2] Bejarano, J. C. R.; Coudert, T.; Vareilles, E.; Geneste, L.; Aldanondo, M.; Abeille, J.: Case-based reasoning and system design: an integrated approach based on ontology and preference modeling. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 28(1), 2014, 49–69. <https://doi.org/10.1017/S0890060413000498>
- [3] Chandrasegaran, S. K.; Ramani, K.; Sriram, R. D.; Horváth, I.; Bernard, A.; Harik, R. F.; Gao, W.: The evolution, challenges, and future of knowledge representation in product design systems, *Computer-Aided Design*, 45(2), 2013, 204-228. <https://doi.org/10.1016/j.cad.2012.08.006>.
- [4] Chen, S.; Yi, J.; Jiang, H.; Zhu, X.: Ontology and CBR based automated decision-making method for the disassembly of mechanical products, *Advanced Engineering Informatics*, 30(3), 2016, 564-584. <https://doi.org/10.1016/j.aei.2016.06.005>
- [5] El-Sappagh, S. H.; Elmogy, M.: Case based reasoning: Case representation methodologies, *International Journal of Advanced Computer Science and Applications*, 6(11), 2015, 192-208. <https://doi.org/10.14569/IJACSA.2015.061126>
- [6] Farhan, U.; Tolouei-Rad, M.; Osseiran, A.: Indexing and retrieval using case-based reasoning in special purpose machine designs, *The International Journal of Advanced Manufacturing Technology*, 92(5-8), 2017, 2689–2703. <https://doi.org/10.1007/s00170-017-0274-5>
- [7] Fenves, S. J.; Fofou, S.; Bock, C.; Sriram, R. D.: CPM2: A core model for product data, *Journal of Computing and Information Science in Engineering*, 8(1), 2008, 014501/1-6. <https://doi.org/10.1115/1.2830842>
- [8] Gao, J.; Deng, G.: The Research of Applying Domain Ontology to Case Based Reasoning System, *Proceedings of International Conference on Services Systems and Services Management*, Chongqing, China, 2005, 1113-1117. <https://doi.org/10.1109/ICSSSM.2005.1500169>
- [9] Guo, Y.; Hu, J.; Peng, Y.: A CBR system for injection mould design based on ontology: a case study, *Computer-Aided Design*, 44(6), 2012, 496-508. <https://doi.org/10.1016/j.cad.2011.12.007>
- [10] Hashemi, H.; Shaharoun, A. M.; Sudin, I.: A case-based reasoning approach for design of machining fixture, *The International Journal of Advanced Manufacturing Technology*, 74, 2014, 113-124. <https://doi.org/10.1007/s00170-014-5930-4>
- [11] Horrocks, I.; Patel-Schneider, P. F.; Boley, H.; Tabet, S.; Grosz, B.; Dean, M.: SWRL: A Semantic Web Rule Language Combining OWL and RuleML, <http://www.w3.org/Submission/SWRL>, 2004
- [12] Hu, J.; Qi, J.; Peng, Y.: New CBR adaptation method combining with problem–solution relational analysis for mechanical design, *Computers in Industry*, 66, 2015, 41–51. <https://doi.org/10.1016/j.compind.2014.08.004>
- [13] Jeh, G.; Widom, J.: SimRank: A Measure of Structural-Context Similarity, *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2002, 538-543. <https://doi.org/10.1145/775047.775126>
- [14] Lee, K. S.; Luo, C.: Application of case-based reasoning in die-casting die design, *The International Journal of Advanced Manufacturing Technology*, 20, 2002, 284–295. <https://doi.org/10.1007/s001700200154>
- [15] Musen, M. A.: The Protégé project: A look back and a look forward, *AI Matters*, 1(4), 2015, 4-12. <https://doi.org/10.1145/2557001.25757003>
- [16] O'Connor, M.; Das, A.: SQWRL: A Query Language for OWL, *OWLED'09 Proceedings of the 6th International Conference on OWL*, Chantilly, VA, 2009, 208-215.
- [17] Qi, J.; Hu, J.; Peng, Y.-H.; Wang, W.; Zhang, Z.: A case retrieval method combined with similarity measurement and multi-criteria decision making for concurrent design, *Expert*

- Systems with Applications, 36(7), 2009, 10357–66.
<https://doi.org/10.1016/j.eswa.2009.01.042>
- [18] Shiu, S. C. K.; Pal, S. K.: Case-based reasoning: Concepts, features and soft Computing, Applied Intelligence, 21(3), 2004, 233–238.
- [19] Singhal, A.: Introducing the Knowledge Graph: Things, Not Strings, 2012, <https://www.blog.google/products/search/introducing-knowledge-graph-things-not/>
- [20] Wang, H.; Rong, Y. K.: Case based reasoning method for computer aided welding fixture design, Computer-Aided Design, 40(12), 2008, 1121–1132.
<https://doi.org/10.1016/j.cad.2008.11.001>
- [21] Watson, I.; Marir, F.: Case-based reasoning: A review, The Knowledge Engineering Review, 9(4), 1994, 327–354. <https://doi.org/10.1017/S0269888900007098>
- [22] Prud'hommeaux, E.; Seaborne, A.: SPARQL Query Language for RDF, 2008, <https://www.w3.org/TR/rdf-sparql-query/>
- [23] Yang, D.; Powers, D. M.: Measuring Semantic Similarity in the Taxonomy of WordNet, Proceedings of the Twenty-eighth Australasian Conference on Computer Science, Newcastle, 2005, 315–322.
- [24] Zhang, J.; Wang, F.; Wang, C.: Integrating case-based with rule-based reasoning in body-in-white fixture design, The International Journal of Advanced Manufacturing Technology, 85(5–8), 2016, 1807–1824. <https://doi.org/10.1007/s00170-015-8040-z>
- [25] Zhang, Y.; Luo, X.; Zhang, B.; Zhang S.: Semantic approach to the automatic recognition of machining features, The International Journal of Advanced Manufacturing Technology, 89(1–4), 2017, 417–37. <https://doi.org/10.1007/s00170-016-9056-8>
- [26] Zhong, J.; Zhu, H.; Li, J.; Yu, Y.: Conceptual Graph Matching for Semantic Search, Proceedings of the 10th International Conference on Conceptual Structures, Springer, 2002, 92–106.