Similarity Metrics Applied to Graph Based Design Model Authoring

Srinivasan Anandan[1] and Joshua D. Summers[2]

[1]Clemson University, sananda@clemson.edu
[2]Clemson University, joshua.summers@ces.clemson.edu

ABSTRACT

Model reuse is typically facilitated by search and retrieval tools, matching the sought model with models in the case base or database. This paper looks at providing assistance to users authoring design exemplars, a new data structure to represent parametric and geometric design problems, by providing alternative configurations. Four distinct similarity metrics are proposed for model retrieval in an interactive modeling environment: entity similarity, relationship similarity, attribute similarity, and structural similarity. A brief review of graph matching literature has been provided, followed by a technical session showing the application of a set of derived similarity metrics.

Keywords: design exemplar, CAD query language, retrieval, interrogation

1. INTRODUCTION

In engineering design, as with many other disciplines, knowledge or model reuse is sought to reduce workload and development effort. Model reuse is typically facilitated by search and retrieval tools, often matching sought models based upon "similarity metrics". For example, a large tire manufacturing firm in North America designs mold inserts for its tire treads. The tooling cost for each mold insert is approximately 2000$. Hence, in order to reduce the manufacturing cost, the designers retrieve inserts that are similar to the desired shape geometry in that, which fall within a specified tolerance envelope. Different strategies have been employed to retrieve similar models [13, 19]. For example, shape-based search methods are used to retrieve 3-D models and sketches in which spherical harmonics are used in creating discriminating similarity measures [13]. An Engineering Advisory System (EAS) has been developed, which involves the retrieval of knowledge associated with 3-D models [19]. The system takes a 3-D shape as a query and converts it into two simpler "fingerprint representations": *feature vectors* and *skeletal graph*. Feature vectors represent the global shape or geometry of the object, whereas the skeletal graphs minimally represent the topology and the local geometry of the object. Examples of feature vectors include the Geometric Moment Invariants and Geometric Ratios. Search for similar models can be performed through a combination of these two representations.

This paper examines similarity in engineering model authoring contexts, offers a set of metrics that might be employed in graph based model authoring, and demonstrates these metrics by an illustrative case study. Specifically, this approach is applied to the authoring of the design exemplar, a standard representation of mechanical engineering design problem knowledge [39]. These metrics are used in a system that offers the model developer, in this case the exemplar developer, options of existing models that are matched to what has been partially authored.

As mentioned above, the design exemplar is a new data structure that has been developed to represent parametric and geometric design problems [7]. Most queries associated with mechanical engineering design include parametric and geometric variables associated with some characteristic of interest for the designer [8]. Entities and their relationships, found explicitly or implicitly in the design model, describe these characteristics. Design exemplars represent these characteristics as patterns, or sub-graphs, of topologic (such as boundary relations, edges, faces), geometric (such as points, lines, planes, angle relations), and algebraic (such as equality and inequality relations and parameters) entities and relationships. Users of the design exemplar create these graphs to interrogate (query) or modify (transform) computer aided design (CAD) solid models. This research is aimed at providing assistance in authoring exemplars for large, complex design problems. Essentially, the objective is to give exemplar authors alternative possible configurations based upon what they have created thus far by searching extensive library sets of exemplars that have been previously built. Thus, the user can simply make changes to the existing exemplars rather than by constructing the complete exemplar from scratch. From the options provided, the author can choose the exemplar that is most useful.

2. THE DESIGN EXEMPLAR

The design exemplar is a data structure that, when combined with a generic constraint solving algorithm, is a construct representing geometric and parametric design problems. Designers, using exemplars, can find geometric properties (e.g. walls with a specific thickness) in CAD models and then change these properties as needed (e.g. the wall thickness from 0.1" to 0.5"). For example, a designer searching for a circle with a radius between 1" and 3" may create the exemplar illustrated in Fig. 1 and shown in text format in Fig. 2.

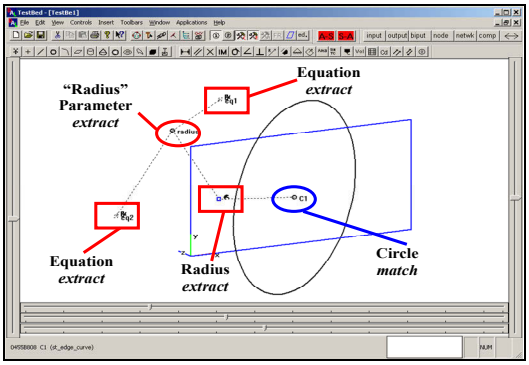


Fig. 1. Circle Radius Checking Exemplar.

Match:
Circle "C1";
Extract:
Parameter "radius";
Radius (radius, C1);
Equation "eq_1" (radius);
Equation "eq_2" (radius);

eq_1: radius < 3
eq_2: radius > 1

Fig. 2. Circle Exemplar.

The circle entity is designated *match*, meaning that the design exemplar entity must be matched in a model. The radius constraint relating the radius parameter and the circle entity is used to extract the radius value from the matched circle in the model. This radius parameter is then evaluated using two equations (radius > 1 and radius < 3). The radius parameter, radius constraint, and two equations are designated as *extract* since these entities and relations are not actually found in the model, but are used by the exemplar algorithm to determine the geometric/parametric aspects of the matched circle. At this point, this design exemplar can be applied against various CAD models, composed of circles, lines, planes, or any other geometric entities, to determine if any of them have circles with radii between 1 and 3. One aspect of the exemplar not discussed here is the transformative power that can modify characteristics by adding, changing, or deleting information in the models. For a complete discussion on the design exemplar, capabilities and limitations, see [34, 39].

A more sophisticated illustration of an exemplar showing more utility represents a design guideline stating that tapers from a parting line should be used to improve the quality of cast parts [24]. Fig. 3 shows a simple part model, with a parting line (plane) without tapers. Based on the guideline, this part would be improved as seen in Fig. 4 where two of the four possible surfaces that intersect the split line are tapered. To do so, the first step is to determine the possible faces that should be tapered. Fig. 5 shows a design exemplar in text format written to find the surfaces that should be tapered based on the design guideline. These surfaces are identified as intersecting the parting line (plane) as the exemplar matches in the CAD model an identified parting line, a plane, a line bounding that plane, and two points bounding that line. This information is then used to determine (1) the angle between the matched split-line plane and the plane bounding the solid model, (2) a point incident on both the split-line plane and the matched line, and (3) the distance from the extracted point to each of the two vertices. Exemplars have been used in feature recognition systems, to model standard design procedures, for manufacturing rule validation, and for view transformation [7, 35, 44].

The process of authoring exemplars can be tedious depending on the type of design problem that is being interrogated [37, 38, 44]. For example, a design problem could be to find all edges in a design model that have length less than one unit. This query will have four entities and five relations. This design problem can be contrasted with a more complicated one for the sizing and configuration of a V-Belt system which includes 53 entities and 58 constraints [38]. Hence, it would be very useful if there were a way to provide some kind of assistance to the author while authoring such complex exemplars. One approach towards easing development effort introduces atomic design exemplars that are networked and compiled into a single larger, more complex exemplar [38]. While this networking strategy increases the reusability of the design exemplar, it does not facilitate the retrieval of existing exemplar models. Authoring an exemplar for a large design problem can take a good deal of effort if a user is starting from scratch, using a generative design approach. Hence, an alternative approach to generative design is variant design, which refers to the technique of adapting existing design specifications to satisfy new design goals and constraints [14]. So, instead of the exemplar author manually looking through the entire case base for similar exemplars, it would be extremely useful

to provide the author with similar exemplars based on the current status of the authoring process. This is analogous to the auto-complete function available in most word processing programs such as when a person may want to type the word 'December'. The moment the writer types the letter string 'dece', an option tab appears on the screen, which says 'December'. On hitting the return key, the word 'December' is completed in the text.

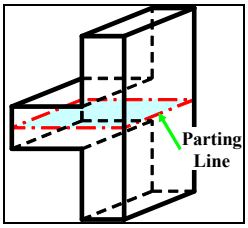


Fig. 3: Model without Tapers

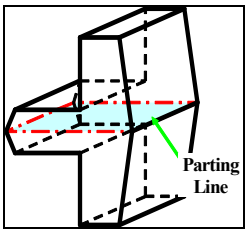


Fig. 4. Model with Tapers.

Match:
- Solid "Body";
- Plane "Splitting Surface";
- Plane "Surface";
- Line "Curve 1";
- Point "Point 1";
- Point "Point 2";
- *ID* "Split-Line Plane" (Splitting Surface);
- *Boundary* (Body, {Surface});
- *Boundary* (Surface, {Curve});
- *Boundary* (Curve, {Point 1, Point 2});

Extract:
- Parameter "angle";
- Parameter "dist_1";
- Parameter "dist_2";
- Parameter "dist_3";
- Point "Point 3";
- *Incident* (Point 3, Curve 1);
- *Incident* (Point 3, Splitting Surface);
- *Angle* (angle, Splitting Surface, Surface);
- *Distance* (dist_1, Point 1, Point 3);
- *Distance* (dist_2, Point 2, Point 3);
- *Distance* (dist_3, Point 1, Point 2);
- *Equation* "eq_1" (angle);
- *Equation* "eq_2" (angle);
- *Equation* "eq_3" (dist_1, dist_2, dist_3);

eq_1: angle < 91 degrees
eq_2: angle > 89 degrees
eq_3: dist_1 + dist_2 = dist_3

Fig. 5. "Casting Taper Rule" Exemplar.

The goal of this paper is to apply the concepts of case-based reasoning [1, 2, 6, 15, 18, 21, 23, 27, 31, 36, 45] and mixed-initiative reasoning [3, 12, 16, 17, 28, 33] to the authoring process for building design exemplar. Case-based reasoning solves new problems by retrieving, and then applying, solutions to previous problems [5, 6]. Mixed-initiative reasoning is a flexible interaction strategy in which each agent (human or computer) makes its best-suited contribution at the most appropriate time. The vision here is to develop an automated and interactive assistant tool, which serves as an assistant to the exemplar author by detecting situations where it can help the author and initiating (tactfully) interaction with him. The first step in developing this tool relates to the similarity measures that may be employed.

3. GRAPH MATCHING CONCEPTS

In this paper, the concept of similarity is being investigated with respect to relational graphs. A relational graph consists of finite number of edges and nodes. A brief summary of the basic concepts of graph matching is provided below based on [10]. A bijective mapping from the nodes of a graph g to the nodes of a graph g^1, which also preserves all labels and the structure of the edges, is defined as a graph *isomorphism* from a graph g to a graph g^1. Another important concept in graph matching is *maximum common subgraph*. A maximum common subgraph of two graphs, g and g^1, is defined as a graph g'' that is a subgraph of both g and g^1, which has the maximum number of nodes as compared to all the possible subgraphs of 'g' and 'g^1'. The concept of graph isomorphism is a useful concept to find out if two objects are the same, or if one object is part of another object, or if one object is present in a group of objects. The concept of *the minimum common supergraph* of two graphs was introduced in [11]. Intuitively similar to the concept of subgraph, a graph g containing two graphs, g^1 and g'', as subgraphs, is defined as a supergraph of g^1 and g''. The minimum common supergraph of g^1 and g'' is a graph that is a supergraph of both g^1 and g'' and has the minimum number of nodes as compared to all those supergraphs.

Instead of computing the maximum common subgraph of two graphs, the *error-tolerant graph matching* can be evaluated using *graph edit distance*. A graph *edit operation* can be a deletion, insertion, or substitution. An example of substitution could be a label change. Both nodes and edges can be subjected to edit operations. The shortest sequence of edit operations needed to transform graph g into graph g^1 is defined as the edit distance of the two graphs. The shorter the edit distance, the more similar two graphs considered to be. In this paper as well, the similarity

between two exemplars is evaluated on the basis of the edit distance. Often, there are *costs* associated with individual edit operations. Typically an edit operation with a more likely occurrence has a smaller cost associated with it. This association of costs with the individual edit operations is often defined in terms of a *cost function*. Thus, a simple definition of graph edit distance computation is to find a set of edit operations that convert one graph to another graph with minimum cost.

A common approach of expressing the concept of similarity or dissimilarity is by means of a distance function [25]. If D is a domain, and x and y are two objects belonging to that domain, then the similarity between them can be expressed by dist(x, y). Often, the distance function is assumed to satisfy the three properties, non-negativity, symmetry, and triangular inequality [41]. Graph isomorphism, subgraph isomorphism, and maximum common subgraph detection can be considered to be instances of graph edit distance computation under special cost functions [9]. As well, weighted graph matching can be regarded a special case of graph edit distance [4, 43].

The concept of graph similarity is independent of the algorithms that have been developed for use in graph matching. Most of the algorithms that have been developed rely on some kind of tree search that make use of various heuristic look-ahead techniques in order to narrow down the search space. The standard algorithm for graph and subgraph isomorphism detection is the one by Ullman [42]. The problem of maximum common subgraph (MCS) detection has been addressed in [20, 22, 26]. In [22] a MCS algorithm that uses a backtrack search is introduced. A different strategy for deriving the MCS first obtains the association graph of the two given graphs and then detects the *maximum clique* (*MC*) of the latter graph [20]. The work in [26] is centered on formulating the maximum clique problem in terms of a continuous optimization problem. A class of continuous and discrete time "replicator" dynamic systems is developed in evolutionary game theory and it is shown how they can be employed in order to solve the relational matching problem.

Classical methods for error-tolerant graph matching can be found in [30, 32, 40]. In [30] the graph distance measures are grouped into two categories.

(1) *Feature-Based Distance,* where a set of features is extracted from the structural representation. These features are then used as an n-d vector where the Euclidean distance can be applied.
(2) *Cost-Based Distances,* where the distance between two objects measures the number of modifications required to convert the first object to the second. In [32], it is stated that an optimal mapping between graphs can hardly be defined. An inexact graph matching algorithm is used to evaluate the structural similarity between graphs.

These methods are guaranteed to find the optimal solution but require exponential time and space due to the NP-completeness of the problem. In this paper, much effort is concentrated on the accuracy of the retrieved exemplars rather than the speed with which the exemplars have been developed. As well, the underlying logic behind the algorithm is to minimize the edit-distance or the number of steps required to change exemplar *a* to exemplar *b*.

4. PROPOSED METRICS

Similarity measures have been widely recognized as key to case retrieval. For the problem domain of design exemplar authoring, four similarity metrics have been suggested, namely entity, relational, structural, and attribute similarity, all of which may be used in combination. The objective is to reduce the number of exemplars in the case base to a manageable number. Hence a series of algorithms were developed to implement these filters.

4.1 Entity Similarity

The retrieval system needs to suggest alternative configurations as the user starts authoring exemplars. First, a filter on all available exemplars is employed that will eliminate those exemplars that do not contain at least as many entities of as many types as specified by the user. The entity similarity measure is defined as the number of entities that are shared between the query and the case. For example, three exemplars A, B, and C can be considered, the entities of which can be represented as three sets. *Set A: (p, q, r, s); Set B: (p, q, r); Set C: (p, r).* Taking the size of the intersection of the sets (SIZE{*Set A* ∩ *Set B*}*=3;* and SIZE{*Set A* ∩ *Set C*}*=2*) a definition of similarity in terms of the entities may be derived. In this case, exemplar A is more similar to B than to C since the number of common entities shared by A and B is more than the number of entities between A and C. This metric serves as the first level of parsing the exemplars. So, for example, it can be considered that the user in the process of authoring an exemplar has thus far authored three circles. On application of this filter, all the exemplars in the case base that do not have at least three circlers are filtered out of the case base. This poses a limitation on the filtering of the exemplar. The assumption behind having this filter is that the exemplar author would not wish to look at design exemplars having less than three circles. For example, it is possible that there may be an exemplar in the case base which is similar to the one that the exemplar author is trying to author except that it has two circles instead of three. The author would be able to use this

exemplar by adding one entity. The application of this filter will however filter out this exemplar. As well, it is possible that the exemplar author authors three edge curves. A circle is a type of an edge-curve, but the application of the entity filter would not return such exemplars.

4.2 Relation Similarity

Second, a filter will be employed that will eliminate those exemplars that do not contain at least as many relations of as many types as specified by the user. Similar to the entity similarity metric, the relation similarity measure is defined as the number of relations that are shared between the query and the case. For example, in addition to the three circles the user decides to author two 'tangent' relations. On application of this filter, of those exemplars that are left in the case base after the first filter, those exemplars that do not have at least two 'tangent' relations are filtered out of the case base. In case equality relations are present in the exemplar being authored, all exemplars not having equality relations are filtered out on application of this filter. However, the filter does not evaluate the similarity of the equations. However as part of future research an investigation of similarity evaluation of the equations will be made.

4.3 Attribute Similarity

Having employed the above mentioned similarity metrics, the third measure is based upon the attributes (alpha, alpha-beta, beta, match, and extract) of the entities and relations. Inclusion of this metric is important, since the number of changes made to exemplar 'a' to change it to exemplar 'b' includes the changes made to the attributes of the entities and relations. The attribute 'match' corresponds to characteristics that are explicit in the design model, whereas the attribute 'extract' refers to characteristics that are implicit in the model. The attributes alpha and beta correspond to the states before and after modification of some characteristics of the model, whereas alpha-beta corresponds to characteristics that are present in the model both before and after modification. The attribute filter is employed to eliminate those exemplars that do not have entities and relations with the same attributes as authored by the user. Thus, for example, the circles in the above exemplar are *alpha-match*. On application of this filter, those exemplars that have three circles but the circles are not *alpha-match* were filtered out of the case base. The limitation to this algorithm is the same as that for the entity similarity metric. For example, if there is an exemplar that is similar to the one that the user is trying to author except that the attributes of the circles are not alpha-match, then the filter would filter out this exemplar.

4.4 Structural Similarity

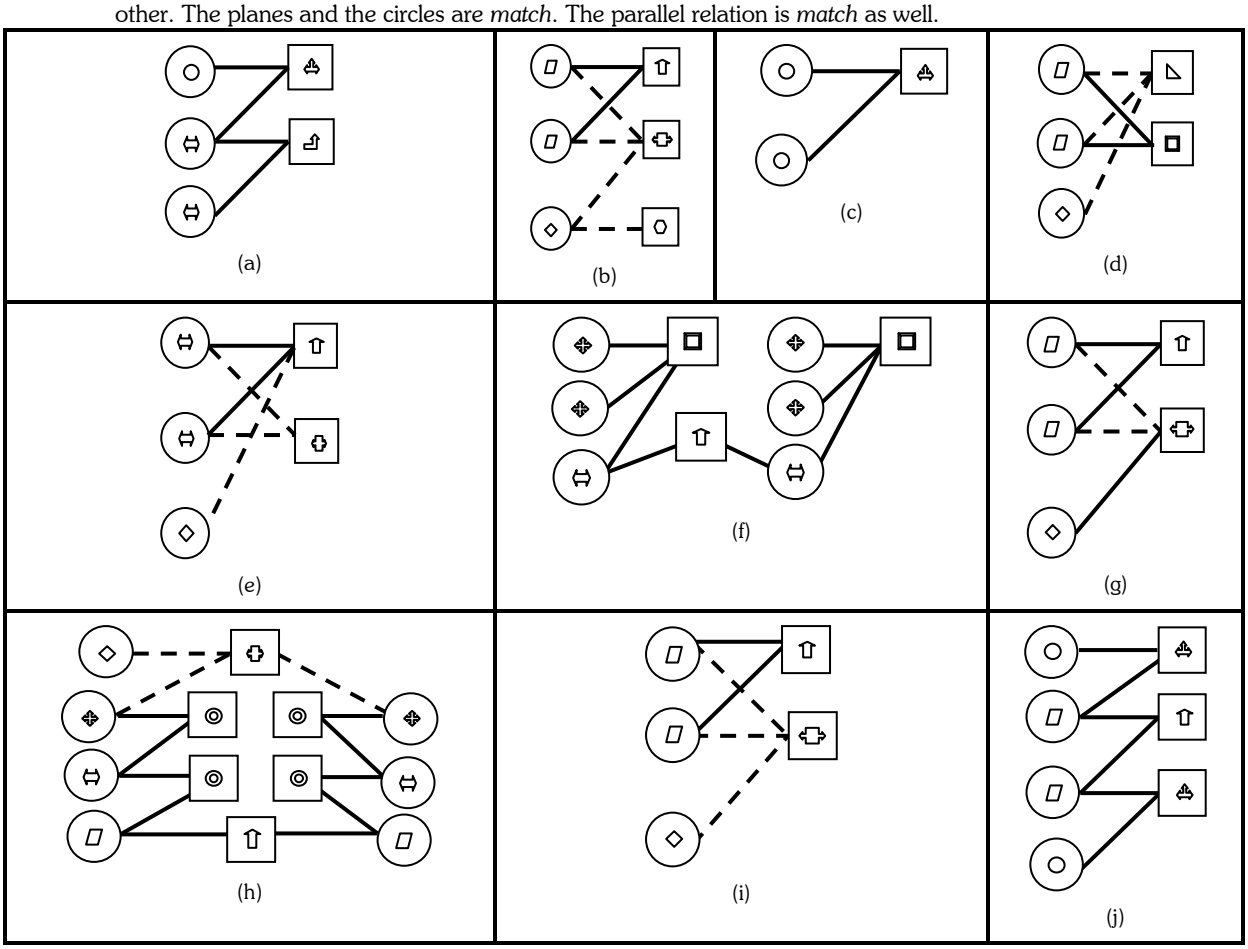
There is a possibility that even after the application of these filters, there will be a sizeable number of exemplars left in the case base. To determine which among these exemplars are most similar to the exemplar that is being authored, it is necessary to determine how the entities and relations are structurally related. Hence a filter is employed to screen those exemplars that are not structurally similar to the one that is being authored. To do so, the existing pattern matching algorithm is used. The current exemplar is passed as a query to the exemplars in the case base. On finding a match, that exemplar is loaded in a separate window for viewing. If no match is found then the query is refined by removing the relations one by one, as each time the refined exemplar is supplied as a query to the pattern-matching algorithm. Once a match is found, it is loaded in a separate window. However, if no matches are found even at this stage, then the query is refined by removing two relations at a time. In case, if there were more than twenty matches at any point, then the exemplars with least number of entities and relations in total are displayed. The following theoretical session explains in detail the application of the filters mentioned above.

5. THEORETICAL SESSION

This section is aimed at demonstrating the application of all the metrics mentioned above, with the help of a sample case base. Tab. 1 shows the cases in the case base. Each exemplar shown has been described:

a. Finds all models that have a circle, a line tangent to the circle, and another line perpendicular to that line. All the entities and relations are *match*.
b. Finds all models that have two planes parallel to each other. The planes and the parallel relation are *match* in the design model. Having found parallel planes, the exemplar finds the distance, checking if the distance is less than 10 units. The parameter entity and distance and equation relations are *extract*.
c. Finds all models that have two circles that are tangent to each other. All the entities and relations are *match*.
d. Finds the angle between all the planes in the model that share a boundary. The plane entities and the boundary relation are *match*, while the angle relation is *extract*.
e. Finds the distance between all pairs of lines that are parallel to each other. The lines and the parallel relation are *match*. The distance parameter and the distance relation are *extract*.

f. Finds all models that have two lines parallel to each other. The lines are bound to two points by the boundary relation. All the entities and relations are *match*.
g. Finds the distance between pairs of parallel planes in the model, as all entities and relations are *match*.
h. In this exemplar, two points are coincident on two lines respectively, which in turn are coincident on two planes that are parallel to each other. The distance relation finds the distance between the two points. The distance relation and the distance parameter are *extract*, while other entities and relations are *match*.
i. Finds the distance between pairs of parallel planes in the model, but the distance relation and the parameter entity are *extract*. The planes and the parallel relation are *match*.
j. Finds all models that have two circles tangent to two planes respectively which in turn are parallel to each other. The planes and the circles are *match*. The parallel relation is *match* as well.



Tab. 1. Sample Case-Base.

Fig. 6 shows the legend for the symbols used in Tab. 1. As an example, a design exemplar user is trying to author an exemplar to find pairs of parallel planes in a model, the distance between which is less than 25 units and more than 10 units. The exemplar that is the objective for the exemplar modeler is found in Fig. 7. This exemplar will consist of two planes constrained by the parallel and distance relations. It will also consist of two equality relations to constrain the value of the distance.

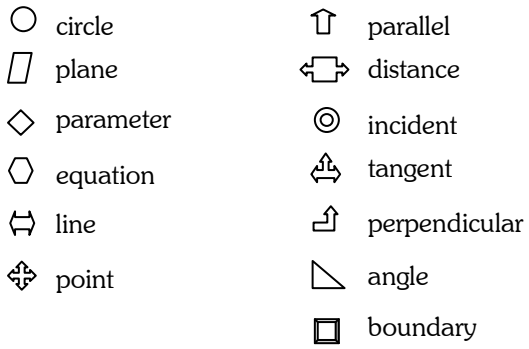


Fig. 6. Legend for the exemplars in Tab. 1.

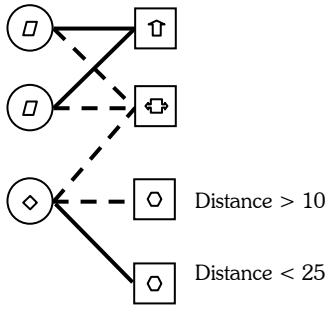


Fig. 7. Designer's Idealized Design Exemplar.

Fig. 8 shows the state when the user has authored two plane entities and the parallel relation between them. The solid lines imply that the attributes of the entities and the relation are *match*. If the user decides to retrieve exemplars from the collection of existing design exemplars, the similarity metrics are evaluated.
The *Entity Similarity* metric filters out exemplars a, c, e, and f, since these exemplars do not contain at least two planes (Fig. 9). The *Relation Similarity* metric filters out exemplar d, as it does not contain at least one parallel relation (Fig. 10). Further, the attributes of exemplars b, g, h, i, and j match those of the exemplar authored by the user since the planes and the parallel relation are *match*. Finally, the planes and the parallel relation in these exemplars are structurally similar to the planes and the parallel relation in the exemplar being authored. Hence, the *Attribute Similarity* filter and the *Structural Similarity* filter do not filter out any exemplars. After the application of all the filters, exemplars b, g, h, i, and j are left in the exemplar collection (Fig. 11).

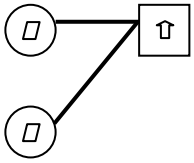


Fig. 8. Initial State of Design Exemplar Authoring Session.

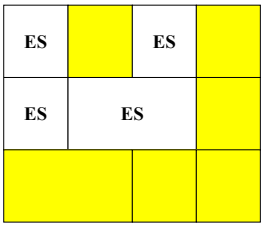


Fig. 9. Application of Entity Similarity Filter (ES).

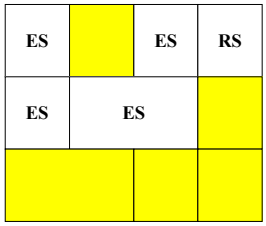


Fig. 10. Application of Relational Similarity Filter (RS).

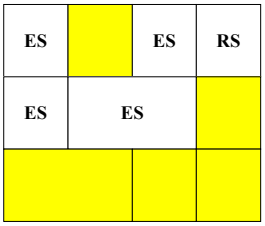


Fig. 11. Exemplars that Are Similar to Initial Exemplar.

As the next step, the user decides to add the distance relation between the parallel planes. If the user decides to get help from the computer at this point, the filters will be again applied as mentioned in the first step. Figure 11 shows the state when the user adds the distance relation. Since exemplars b, g, and i have at least one 'distance' relation and it is 'extract', these exemplars get retained in the case base. As well, since exemplar h is not structurally similar to the one being authored, it gets filtered out (Fig. 13). In the next step, the user adds the 'parameter' entity and makes it 'extract' (Fig. 13). At this instance, of the exemplars left in the case base, exemplars b and i match the exemplar authored by the user, since the parameter entity in exemplar g is not extract. When these exemplars get displayed, the user decides to accept exemplar b, since there is only one more step needed to attain the user's objective (Fig. 15).

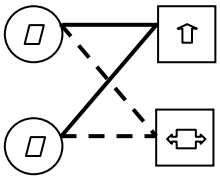


Fig. 12. State of Design Exemplar Authoring Session.

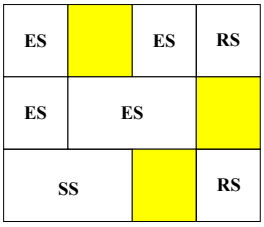


Fig. 13. Exemplars Similar to Fig. 11.

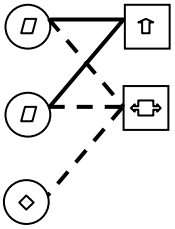


Fig. 14. State of Design Exemplar Authoring Session.

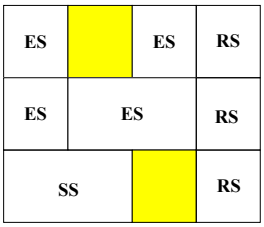


Fig. 15. Exemplars Similar to Fig. 12.

Studies in the field of cognitive psychology have shown that at any point of time, humans can best process information, if it is presented to them in the form of 7 ± 2 options [29]. Hence it was decided that the user should be provided not more than 7 options at any point of time. To ensure that this feature is not a hindrance to the normal functioning of the exemplar, it was decided to provide a button, "RETRIEVE", which will do the above procedure once each time the user, clicks the button. That way the user has the choice of using this feature as and when he desires. As well, of all the potential matches, only those exemplars were left in the case base of which the current exemplar forms 100% and less than 75%. The thought behind this was that the user, most likely, will be looking for help in the immediate steps following that point in time, and is not looking too ahead in the future. If the lower limit would be any lesser than 75%, then there is possibility that the user would have to look too much ahead of time. The reason for these numbers is that if it is 100%, then that exemplar has no use to the author. In other words, he has authored a replicate.

6. DISCUSSION

The technical session demonstrates that the results obtained through the application of the algorithms can be satisfactory. However there are certain limitations to the algorithm that has been developed. For instance, there could be different ways to author an exemplar for a specific purpose. For example, while authoring an exemplar consisting of two circles and two planes, the user could either start with two circles and add the remaining entities and relations or start with two planes and then add the other entities and constraints. If the user decides to seek the computer's help after adding the first two entities to the exemplar, then the results obtained could be different in both cases. Such situations emphasize the need for a retrieval system to capture the purpose of the exemplar that is being authored.

Consider, for example, that the user wants to author an exemplar to find the distance between two parallel planes. There are a various ways to author an exemplar for this purpose. The user could define two planes, and the distance relation can relate the planes. The exemplar author could also define two planes, define two lines incident on the planes, make two points incident on the line and then the distance relation can relate the two points. In both cases, the author will be able to retrieve the distance between the two planes. However, irrespective of whatever method the author chooses, the algorithm will not be able to retrieve the other exemplar when the author decides to use the retrieval feature. So, there is a semantic gap that may exist between the exemplar being authored and the one that is retrieved.

As well, because the filters work based on the entities and relations already existing in current exemplar, there is a possibility that an exemplar may be filtered even if it has one less entity or relation. For example, it can be considered that the user is trying to author an exemplar having three circles and two tangent relations. The entity and relation filters would filter even those exemplars that may have two circles, two tangent relations and one line. So, in other words, the algorithm assumes that the author does not wish to delete any entity or relation once he adds it. These limitations will be addressed in the future.

6.1 Future Work

As mentioned above there are some limitations that need to be addressed. Different approaches have been identified as potential directions to make this algorithm more effective. One method could be to assign more flexibility to the retrieval algorithm. Letting the authors decide on the filters they need or feel most useful could do this. Case studies can be conducted to determine the users' preference in the sequencing of the filters. This could prove to be a better way to address the individual needs of each user and thus make the retrieval feature more useful.

Addressing the semantic gap between the exemplar that is being authored and the ones that are retrieved is one more aspect of the retrieval feature that can make it more useful. As mentioned above, there are different approaches an exemplar can be authored to achieve a particular purpose. A mixed-initiative reasoning system, in the real sense of the term, could be in place if the computer is able to provide the user with alternative and less complex ways to author the same exemplar. This can be done if the algorithm is able to capture the purpose of the exemplar that the user is trying to author. If we consider the example of finding the distance between two parallel planes mentioned above, then the algorithm should be able to provide the user with the simpler options based upon the purpose of the exemplar, in this case, finding the distance between two parallel planes.

7. REFERENCES

[1] Aamodt, A., Plaza, E., Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches, *AI Communications*, Vol. 7, 1994, pp 39-59.

[2] Aitken, S., Case-Based Reasoning at AIAI: CBD Shell v.2, *url*, http://www.aiai.ed.ac.uk/project/cbr, www.aiai.ed.ac.uk/projct/cbr, 2002, accessed: November 15, 2003.

[3] Allen, J. F., Guinn, C. I. ,Horvitz, E., Mixed-Initiative Interactions, *IEEE Intelligent Systems*, Vol. 14, No. 5, 1999, pp. 14-23.

[4] Almohamed, H., A Linear Programming Approach for the Weighted Graph Matching Problem, *IEEE Transactions on PAMI*, Vol. 15, 1993, pp. 522-5.

[5] Amen, R., Vomacka, P., Case-Based Reasoning as a Tool for Materials Selection, *Materials and Design*, Vol. 22, 2001, pp. 353-358.

[6] Bardasz, T., Zeid, I., DEJAVU: A Case-Based System to Aid Novice Designers, *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, Vol. 7, No. 3, 1993, pp. 111-124.

[7] Bettig, B., Shah, J. J. ,Summers, J. D., Domain Independent Characterization of Parametric and Geometric Problems in Embodiment Design, *Design Engineering Technical Conferences*, ASME, Vol. DETC-2000, Baltimore, MD, 2000, DAC-14259.

[8] Bettig, B., Shah, J. J., Summers, J. D., Geometric Exemplars: A Bridge Between AI and CAD, in *From Knowledge Intensive CAD to Knowledge Intensive Engineering*, Cugini and Wozny (eds), Kluwer Academic Press, the Netherlands, 2001.

[9] Bunke, H., Error Correcting Graph Matching: On the Influence of the Underlying Cost Function, *IEEE Transactions PAMI*, Vol. 21, 1999, pp. 917-22.

[10] Bunke, H., Graph Matching: Theoretical Foundations, Algorithms, and Applications, *Vision Interface 2000*, Montreal, Canada, 2000, 82-8.

[11] Bunke, H., Jiang, X., Kandel, A., On the Minimum Common Supergraph of Two Graphs, *Computing*, Vol. 65, No. 1, 2000,

[12] Burstein, M. ,Mcdermott, D., Issues in the Development of Human-Computer Mixed-Initiative Planning Systems, in *Cognitive Technology: In Search of a Humane Interface*, Gorayska and Mey (eds), Elsever Science B.V., 1996.

[13] Funkhouser, T., Min, P., Kazdan, M., Chen, J., Halderman, A., Dobkin, D., Jacobs, D., A Search Engine for 3D Models, *ACM Transactions on Graphics*, Vol. 22, No. 1, 2003,

[14] Gero, J., Computational Models of Creative Design Processes, in *AI in Creativity*, Dartnall (eds), Kluwer, Dordrecht, the Netherlands, 1994.

[15] Goel, A., Bhatta, S., Stroulia, E., KRITIK: An Early Case-Based Design System, in *Issues and Applications of Case-Based Reasoning in Design*, Maher and Pu (eds), Erlbaum, Mahwah, NJ, 1997.

[16] Hartrum, T., Deloach, S., Design Issues for Mixed-Initiative Agent System, *AAAI-99 Workshop on Mixed-Initiative Intelligence*, AAAI Press, Menlo Park, CA, 1999, pp 40-44.

[17] Hearst, M., Mixed-Initiative Interaction Commentary, *IEEE Intelligent Systems*, Vol. 14, No. 5, 1999, pp 14-24.

[18] Hinrichs, T., Some Limitations in Feature-Based Recognition in Case-Based Design, *ICCBR-95*, Sesimbra, Portugal, 1995, pp 471-80.

[19] Iyer, N., Patwardhan, H., Jayanti, S., Ramani, K., Dynamic Early Design Advice Using Shape Retrieval, *International CIRP Design Seminar*, CIRP, Grenoble, France, 2003,
[20] Levi, G., A Note on the Derivation of Maximal Common Subgraphs of Two Directed or Undirected Graphs, *Calcolo*, Vol. 9, 1972, pp 341-54.
[21] Maher, M., Balachandran, M., Zhang, D., *Case-Based Reasoning in Design*, Lawrence Erlbaum Associates, Mahwah, NJ, 1995.
[22] Mcgregor, J., Backtrack Search Algorithms and the Maximal Common Subgraph Problem, *Software-Practice and Experience*, Vol. 12, 1982, pp 23-34.
[23] Mileman, T., Knight, B., Petridis, M., Cowell, D. ,Ewer, J., Case-Based Retrieval of 3-Dimensional Shapes for the Design of Metal Castings, *Journal of Intelligent Manufacturing*, Vol. 13, 2002, pp. 39-45.
[24] Pahl, G., Beitz, W., *Engineering Design: A Systematic Approach*, Springer-Verlag, New York, NY, 1996.
[25] Papadopoulos, A., Manolopoulos, Y., Structure-Based Similarity Search with Graph Histograms, *DEX/IWOSS International Workshop on Similarity Search*, IEEE Computer Society, 1999, pp 174-178.
[26] Pelillo, M., A Unifying Framework for Relational Structure Matching, *14th ICPR*, Brisbane, 1998,
[27] Perera, S. and Watson, I., NIRAMI: An Integrated Case-Based System for Strategic Design and Estimating, in *Progress in Case-Based Reasoning*, Watson (eds), Springer-Verlag, Salford, UK, 1996.
[28] Ramakrishnan, N., Capra, R. G. and Perez-Quinones, M. A., Mixed-Initiative Interaction = Mixed Computation, *ACM SIGPLAN Notices*, Vol. 37, No. 3, 2002, pp. 119-130.
[29] Sanders, M. and Mccormick, E., *Human Factors in Engineering and Design*, McGraw-Hill, St. Louis, MO, 1993.
[30] Sanfeliu, A. and Fu, K., A Distance Measure Between Attributed Relational Graphs for Pattern Recognition, *IEEE Transactions SMC*, Vol. 13, 1983, pp 353-63,
[31] Schaaf, J., Fish and Shrink: A Next Step Towards Efficient Case Retrieval in Large Scaled Case Bases, in *Advances in Case-Based Reasoning*, Smith and Faltings (eds), Springer-Verlag, Lausanne, 1996.
[32] Shapiro, L., Haralick, R., Structural Descriptions and Inexact Matching, *IEEE Transactions on PAMI*, Vol. 3, 1981, pp 504-19.
[33] Shyr, P., Tecuci, G. and Boicu, M., Evaluation of Mixed-Initiative Knowledge Based Development Methods and Tools, *IJACAI-2001: Workshop on Empirical Methods in AI*, Seattle, WA, 2001, pp 47-53.
[34] Summers, J. D., *Development of a Domain and Solver Independent Method for Mechanical Engineering Embodiment Design*, Ph.D., Mechanical and Aerospace Engineering, Arizona State University, Tempe, 2004.
[35] Summers, J. D., Lacroix, Z. and Shah, J. J., Collaborative Mechanical Engineering Design: Representation and Decision Issues, *Le Travail Humain: Modeling Cooperative Activities in Design*, Presses Universitaires, Vol. 8**,** Paris, France, 2001,
[36] Summers, J. D., Lacroix, Z. and Shah, J. J., Case-Based Design Facilitated by the Design Exemplar, *International Conference on Artificial Intelligence in Design*, Kluwer Academic Press, Vol. 7**,** Cambridge, UK, 2002, pp 453-476.
[37] Summers, J. D. and Shah, J. J., Empirical Studies for Evaluation and Investigation of a New Knowledge Representation Structure in Design Automation, *Design Engineering Technical Conferences*, ASME, Vol. DETC-2002**,** Montreal, Quebec, Canada, 2002, CIE-34488.
[38] Summers, J. D. and Shah, J. J., Exemplar Networks: Extensions of the Design Exemplar, *Design Engineering Technical Conferences*, Computers in Engineering, Vol. DETC-2004**,** Salt Lake City, UT, 2004, CIE-57786.
[39] Summers, J. D., Shah, J. J. and Bettig, B., The Design Exemplar: A New Data Structure for Embodiment Design Automation, *Journal of Mechanical Design*, Vol. 126, No. 5, 2004, pp 775-87.
[40] Tsai, W. and Fu, K., Error-Correcting Isomorphisms of Attributed Relational Graphs for Pattern Recognition, *IEEE Transactions on SMC*, Vol. 9, 1979, pp 757-68.
[41] Tversky, A., *Preference, Belief, and Similarity: Selected Writings*, The MIT Press, Cambridge, MA, 2003.
[42] Ullman, J. R., An Algorithm for Subgraph Isomorphism, *Journal of the Association for Computing Machinery*, Vol. 23, No. 1, 1976, pp 31-42.
[43] Umeyama, S., An Eigen decomposition Approach to Weighted Graph Matching Problems, *IEEE Transactions PAMI*, Vol. 10, 1988, pp 695-703,
[44] Venkatamaran, S., Summers, J. D. and Shah, J. J., An Investigation of Integration of Design by Features and Feature Recognition, *Feature Modeling and Advanced Design for Life Cycle Systems*, IFIP, Valenciennes, France, 2001,
[45] Zdrahal, Z., Motta, E., Case-Based Problem Solving Methods for Parametric Design Tasks, in *Advances in Case-Based Reasoning*, Smith and Faltings (eds), Springer-Verlag, Lausanne, 1996.