Representation, Indexing, and Retrieval of Biological Cases for Biologically Inspired Design

Bryan Wiltgen[1], Ashok K. Goel[1,2], and Swaroop Vattam[1]

[1] Design & Intelligence Laboratory, School of Interactive Computing
[2] Center for Biologically Inspired Design,
Georgia Institute of Technology
Atlanta, GA, USA
{bwiltgen,goel,svattam}@cc.gatech.edu

Abstract. Biologically inspired design is an increasingly popular design paradigm. Biologically inspired design differs from many traditional case-based reasoning tasks because it employs cross-domain analogies. The wide differences in biological source cases and technological target problems present challenges for determining what would make good or useful schemes for case representation, indexing, and adaptation. In this paper, we provide an information-processing analysis of biologically inspired design, a scheme for representing knowledge of designs of biological systems, and a computational technique for automatic indexing and retrieval of biological analogues of engineering problems. Our results highlight some important issues that a case-based reasoning system must overcome to succeed in supporting biologically inspired design.

Keywords: Analogical design, analogical reasoning, biologically inspired design, biomimicry, case-based creativity, case-based design.

1 Introduction

Biologically inspired design, sometimes also called biomimicry or bionics, uses analogies to biological systems to generate ideas for the conceptual (or the preliminary, qualitative) phase of design. Although nature has long inspired designers (e.g., Leonardo da Vinci, the Wright brothers), biologically inspired design recently has become an important and widespread movement, pulled by the growing need for environmentally sustainable designs and pushed by its promise to generate creative designs [1-4]. The design of windmill turbine blades based on the tubercles on a humpback whale's flippers [5-7] illustrates the power of biologically inspired design. By taking inspiration from whale flippers, designers were able to improve the shape of wind turbine blades to improve lift and reduce drag, increasing their efficiency.

Biologically inspired design differs from many traditional case-based reasoning tasks because it employs cross-domain analogies. In classical case-based reasoning (CBR) [8,9], the new, input or target problem is so closely related and similar to familiar, known source cases stored in memory that the case memory supplies an almost correct answer to a given problem and the retrieved case need only be

A. Ram and N. Wiratunga (Eds.): ICCBR 2011, LNAI 6880, pp. 334–347, 2011.

"tweaked" to fit the problem. In contrast, biologically inspired design by definition entails "far" analogies from biology to architecture, engineering, computing, and other design domains, e.g., the design of turbine blades based on the shape of whale flippers or the design of a self-cleaning catheter based on the surface of lotus leaves. Thus, biologically inspired design is an excellent context for studying case-based analogies well as case-based creativity.

In this paper we focus on biologically inspired engineering design that engages transfer of the results of biological evolution to engineering problems. Biology and engineering, however, occur at many different scales in both time and space. Further, biologists and engineers use different languages and methods, and generally share little cultural and disciplinary common ground. This presents an interesting set of challenges for research on case-based reasoning. How does one represent, retrieve, and adapt knowledge from a source domain that is as fundamentally different than the target domain as biology is to engineering?

In this paper, first we briefly describe an information-processing account of biologically inspired design processes. Next, we describe a scheme for representing knowledge of designs of biological systems. Then, we apply the technique of redundant discrimination networks for automatic indexing and retrieval of biological designs. We conclude with a discussion on the results of our application, highlighting identified issues and speculating on how to overcome those difficulties.

2 Case-Based Design

Research on case-based has a long and rich history in which many researchers have not only used CBR theories and techniques in design, but also used insights from design to drive research into CBR. Early examples of development of CBR in design include CYCLOPS [10,11], STRUPLES [12,13] ARGO [14] and KRITIK [15-17]. For example, ARGO used rule-based reasoning to transform design plans for designing VLSI circuits to meet functional specifications of new circuits. In contrast, KRITIK integrated case-based and model-based reasoning to produce conceptual designs for engineering devices such as heat exchange devices and electric circuits. If a designer specified a function, F, KRITIK generated a qualitative specification of a structure S, which could accomplish that function. To do so, it stored an inverse mapping (from structure S, to behavior B, to function F) in the form of a structure–behavior–function (SBF) model for each past case. Thus, SBF model provided a functional vocabulary for indexing past design cases so that they could be stored and later retrieved, adapted, or verified. Maher & Gomez [18] provide a survey of some of the early case-based design systems; Maher & Pu [19] provide an anthology of several early papers.

The last two decades saw an explosion of interest in case-based design. Briefly, we identify four major trends in this period. The first trend was to develop interactive CBR design systems that provided access to libraries of design cases but left the task of design adaptation to the user [20,21]. A second trend was to integrate CBR with a wide variety of reasoning methods such as rule-based reasoning and model-based reasoning [22,23], constraint satisfaction [24,25] and genetic algorithms [26,27] in order to create or evolve emergent new designs from the original case base. A third trend was the development of hierarchical case-based reasoning in which design cases were decomposed into sub cases at storage time and recomposed at problem-solving

time [28,29]. A fourth major trend in research on CBR in the nineteen nineties was to develop CBR for a variety of design tasks, such as assembly planning [30], in a wide variety of design domains such as software design [29,31] and design of human-machine interfaces [32]. Goel & Craw [33] survey some of the above developments in case-based design.

More recently, visual CBR has been at the forefront of research. Gross & Do's [34] Electronic Napkin took queries in the form of simple design sketches and retrieved matching design drawings from a design case library. Other work in visual CBR took design drawings generated by vector graphics programs as input and retrieved matching vector graphics design drawings from a diagrammatic case library [35], and used purely visual knowledge to transfer design plans from a known design case to a new design problem [36].

3 Information Processes of Biologically Inspired Design

We study biologically inspired design in the context of an interdisciplinary, senior-level undergraduate course our university offers on biologically inspired design (ME/ISyE/MSE/PTFe/BIOL 4740). Although its contents change slightly every year,

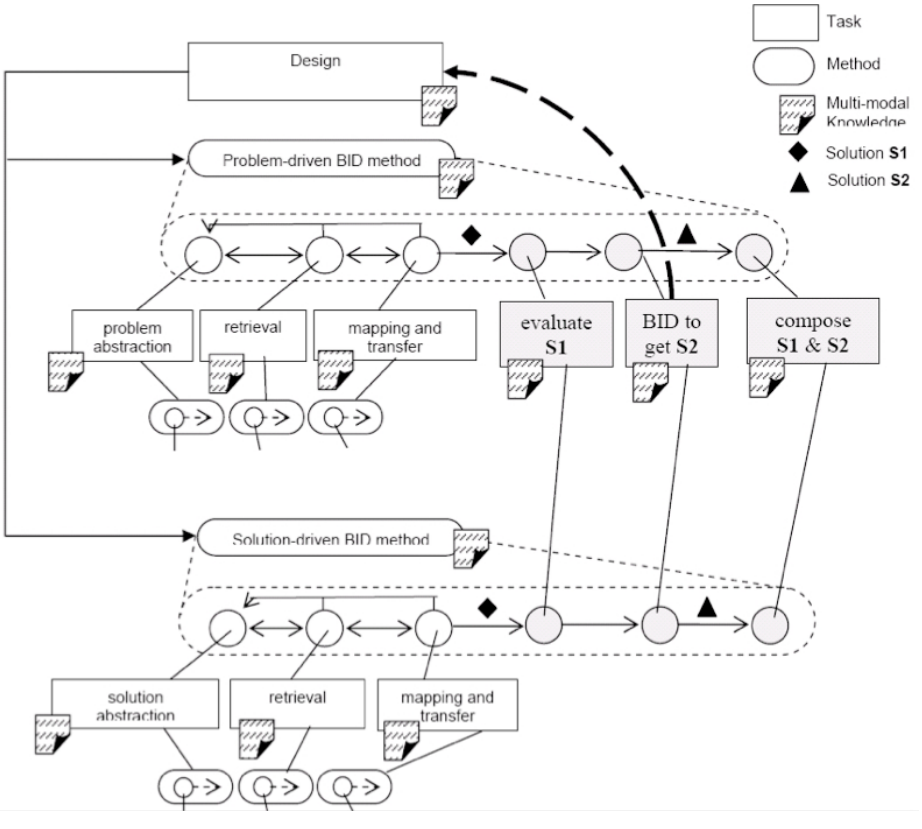

Fig. 1. An Information Processing Model of Biologically Inspired Design (adapted from [37])

it usually consists of three components: lectures, found object exercises, and a semester-long biologically inspired design team project. For the project, teams of 4-6 students are formed such that each team has at least one biology student and students from different schools of engineering. Each team was given a broad problem in the domain of dynamic, adaptable, sustainable housing such as heating or energy use.

Our *in situ* observations have indicated two characteristics of analogical transfer from biology to engineering. First, the process of biologically inspired design can be either problem-driven or solution-driven. The two design processes share many steps but differ in their starting points. As in traditional design, problem-driven biologically inspired design starts with a functional specification of a design problem. In contrast, solution-driven design starts from a biological design looking for a problem to address. The work described here focuses on problem-driven biologically inspired design. Second, biologically inspired design entails compound analogies. In compound analogy, an initial reminding of a biological analog results in a decomposition of the given design problem. The unsolved sub-problems then lead to additional reminders of biological analogues. Figure 1 (adapted from [37]), illustrates our preliminary, high-level information-processing model of biologically inspired design. Note that the ordering of the design subtasks is dependent upon a selection of a particular design method, either problem-driven design (top half of the figure) or solution-driven (bottom half). Note also that due to compound analogies, the process of analogical transfer is iterative and may reoccur for each open sub-problem.

4 Representation of Knowledge of Biological Systems

The promise of biologically inspired design has led to several efforts at developing computational tools for supporting the design process (e.g., [38-40]). In this section, we briefly describe a computational tool called DANE (for Design by Analogy to Nature Engine) that provides a designer with access to a knowledge base of biological systems [41]. To support the problem-driven design process, knowledge of a biological system needs to explicitly specify (1) the functions of the biological system because target design problems typically are specified by the desired functions, (2) the behaviors (or mechanisms) of the biological system that result in the achievement of a specific function because the designer is interested in biologically mechanisms for achieving desired functions, and (3) the structure of the biological system because mechanisms arise from the structure and because the specification of design problems typically contains structural constraints.

Thus, DANE represents designs of biological systems in the SBF knowledge representation language (e.g., [42]). An SBF model of a complex system explicitly represents its structure [S] (i.e., its configuration of components and connections), its functions [F] (i.e., its intended output behaviors), and its behaviors [B] (i.e., its internal causal processes that compose the functions of the components into the functions of the system). The SBF language provides a vocabulary for expressing and organizing knowledge in an $F \rightarrow B \rightarrow F \rightarrow B \ldots \rightarrow F(S)$ hierarchy, which captures functionality and causality at multiple levels of aggregation and abstraction. Generally speaking, as one moves down the hierarchy, a system is described at lower levels of abstraction. For example, one level of the hierarchy might describe, at a

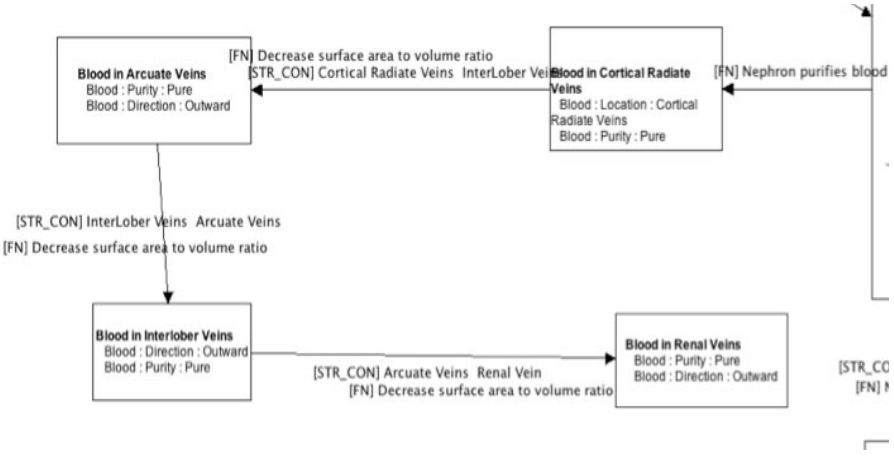

Fig. 2. Partial Behavior Model for the system "Kidney Filters Waste from Blood"

high level of abstraction, the function, structures, and behavior of the entire human digestive system, whereas a deeper level might narrow its focus to describe in detail the function, structures, and behavior of the small intestines.

Biological systems are indexed by function name (e.g., "flamingo filter-feeds self"), by subject (e.g., "flamingo"), and/or by verb (e.g., "filter-feeds"). Upon selecting a system-function pair, users are presented with a multi-modal representation of the paired system-function that combines text descriptions, images, and an SBF model. The function sub-model is given as a structured frame-like representation, and the behavior and structure sub-models are represented as labeled directed graphs. Figure 2 illustrates the behavior (or the mechanism) for the function "Kidney Filters Waste from Blood."

In Fall 2009, we introduced DANE into the class on biologically inspired design mentioned earlier. At the time, the library contained about 40 SBF models, including 22 models of biological systems and subsystems. We discovered that some designers in the class found DANE's SBF models useful for enhancing their conceptual understanding of biological designs. But we also found the designers did not consistently use DANE for generating design concepts for their problems. This probably was due to several reasons such as learning how to use DANE was not easy enough, DANE's library of biological systems was not large enough, DANE did not provide enough design support. For example, in reference to the last item, while DANE allowed the user to browse the library of biological systems, it did not have any ability to automatically index or retrieve biological systems relevant to a target design problem. The interested reader may download DANE at http://dilab.cc.gatech.edu/dane.

5 Automatic Indexing and Retrieval

To help resolve this observed need, we are using analogical reasoning techniques for adding automated indexing and retrieval to DANE. AI techniques for analogical

retrieval include constraint satisfaction [43], spreading activation [44], and redundant discrimination networks [9,45]. Redundant discrimination networks allow a single case to be indexed in multiple networks. We use this scheme for indexing biological designs because it is not known *a priori* what the most apt indexing method is for any given case. Additionally, the Ideal system [46,47] used this scheme with some success. The open question is whether this scheme works for analogical retrieval of biological designs and if it helps further systemize knowledge of biological designs.

Specification of Design Problems

To automatically retrieve biological analogues relevant to a target design problem, we need to first decide on a vocabulary for specifying design problems. Fortunately, the SBF language already provides such a vocabulary. A desired design is defined by three components: the `initial state`, the `objective state`, and a set of `structural connections`. The `initial state` and `objective state` both contain `state features`, which themselves are composed of `object-property-value` triples. A `state feature` may also be flagged as being a `substance`, which means the `object` in that `feature` flows through the system. Each `structural connection` in the set of `structural connections` (the third primary component of a `specification`) links two `objects` (or optionally one `object` to itself) through a `connection type`, which can be any string.

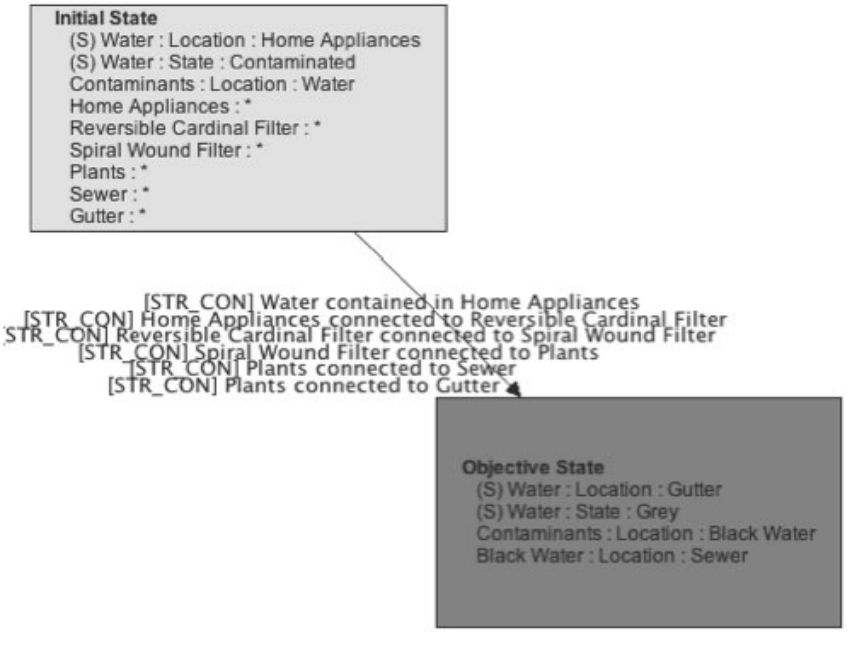

Fig. 3. Specification of a Filtrating Design Problem

As an example, Figure 3 illustrates a specification of the problem of home water filtration taken from a 2009 class project. The `initial state` of this specification describes the `substance` (denoted with (S)) water being contaminated and located within home appliances. The transition between the `initial` and `objective states` describes a series of `structural connections` that are relevant to the filtration process described by the 2009 design. In the `objective state`, the once-contaminated water has been transformed into grey water and moved to the gutter, and its contaminants are now located in black water in the sewer. In Figure 3, note that a state feature like "Plants : *" is used to denote an `object` that appears in a `connection` but does not otherwise have a `property` or `value` associated with it.

Structural Discrimination Network Algorithm

For each behavior model:

1: Create one node N for the set of all unique, non-substance structures in a by-state or by-structural-connection transition.
2: Attach node N as a child of STRUCTURAL-ROOT.
3: Create a node M for each by-structural-connection transition that includes at least 1 non-substance structure.
4: Attach each node M as child of N.
5: Create a node C for each non-substance structure in an M-node.
6: Attach each node C as a child of M where structures are equal.
7: Create a node P for each structure-property pair in a by-state transition.
8: Attach each node P as a child of C where structures are equal.
9: Create a node S to represent the system.
10: Attach node S as a child of P.

Functional Discrimination Network Algorithm

For each behavior model:

1: Create a node N for each substance-property pair in a by-state transition
2: Attach node N as a child of FUNCTIONAL-ROOT.
3: Create a node M for each substance-property-value triple in a by-state transition.
4: Attach each node M as child of N where properties are equal.
5: Create a node O to represent the system.
6: Attach node O as a child of all M nodes.

Fig. 4. Pseudocode for the Indexing Algorithms

Automatic Indexing of Biological Designs

For automated retrieval to work, models of the biological systems in DANE need to be indexed. We use two algorithms adapted from [45], and named Functional Indexing and Structural Indexing, both of which generate separate discrimination networks. Figure 4 provides the pseudocodes for these indexing algorithms. The Functional Indexing algorithm discriminates on `substance`-flagged `objects` and their `properties`, whereas the Structural Indexing algorithm discriminates on `structural connections` and the `properties` of the `objects` involved in those `connections`. These trees are traversed by matching components in the input specification (e.g., Figure 3) to nodes in the tree. Note that, during the creation of the discrimination trees in both cases, if a node is about to be created but already exists in the network, the pre-existing node is used instead. This step prevents the network from having many duplicate nodes (e.g., multiple nodes representing the `property` of "Location").

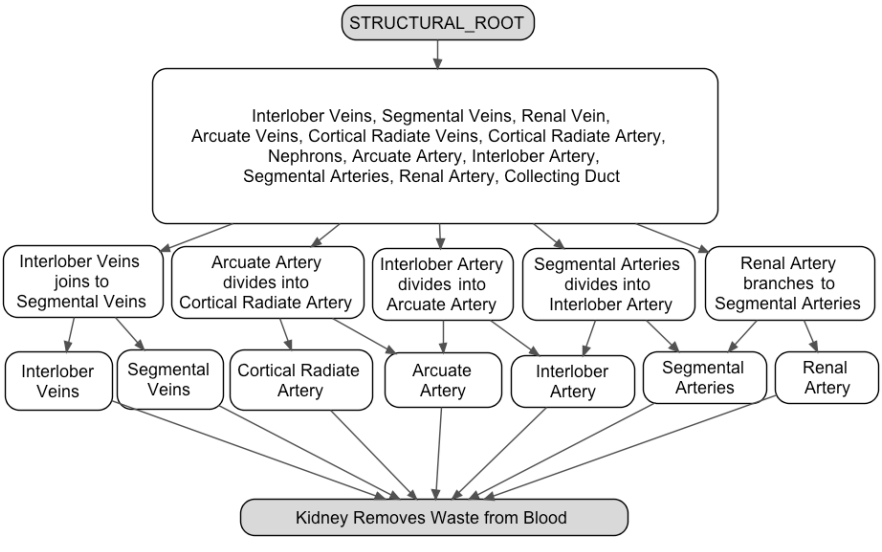

Fig. 5. Partial snapshot for the Structural Discrimination Network

Figure 5 shows a partial discrimination tree built by the Structural Indexing algorithm. At the top level, the root connects to a node that indexes a set of `objects`. Underneath that node is a set of nodes that index on `structural connections`, and underneath them is a set of related `objects`. Finally, the `objects` themselves link to a model in the system. Note that we don't have `properties` in this partial network because the `objects` here did not have specified `properties` in their model.

Automatic Retrieval of Biological Designs

Once a user creates a `specification` and attempts retrieval, the system uses the contents of the `specification` as probes into these discrimination networks and

retrieves one or more results depending upon if the network allows partial matches. We allow partial matches in the Structural network because it is based on non-state specific information (i.e., it does not index on `value`).

Similarity Measurement

If multiple results are returned, the system must determine how similar each result is to the user's `specification` in order to rank the results and recommend a best match. We use two similarity metrics: Structural and Functional. The Structural similarity measurement looks at how many `structural connections` and `object-property` pairs are the same between the `specification` and the returned result. The Functional similarity measurement compares the `initial` and `objective states` of the `specification` and the returned result by attempting to match `object-property-value` triples. This metric differs from the Structural measurement because it compares `values` and does not compare `structural connections`. Both similarity metrics are calculated using Tversky's ratio model [48] with alpha and beta set to 1, which simplifies to $S(specification, case) = f(specification \cap case) / f(specification \cup case)$ with f comparing features relevant to the given similarity measurement.

Every result returned, regardless of which discrimination network was used, gets measured by both similarity metrics. The average of both scores is presented to the user as a percentage of total similarity and used by the system to rank the results.

6 Evaluation

We performed two sets of tests to evaluate automatic indexing and retrieval in DANE. In the first test, our system retrieved of a target biological system from the library. In the second tested, our system retrieved a biological system relevant to a design problem. For both experiments, we developed a library of 37 models. 15 of the 37 models in the library are technological; 22 are biological. Technological models were derived from earlier CBR experiments in our lab and are related to acid coolers, pulleys, gyroscopes, crankshafts, and latches. Biological models were developed by members of our lab (including the authors) with the goal of accurately representing their respective topics and are related to three families of systems: transpiration, the human small intestines, and the human kidney.

The functional discrimination network indexes 4 models: 3 technological and 1 biological. The structural discrimination network indexes 19 models: 8 technological and 11 biological. The Structural discrimination network contains all of the models indexed by the Functional discrimination network. We found that some of 37 models were not indexed by either algorithm because they are legacy models developed prior to our building of this automated retrieval technique, and thus do not contain the knowledge needed by the retrieval algorithms.

Method of the First Test

For the first test, we tested our system's ability to retrieve a target biological system. The target was "Kidney Removes Waste from Blood," a model that describes how the human kidneys filter certain materials out of our blood.

The test involved 28 sub-tests across five phases. For each test, we built a `specification` and asked the system to retrieve a set of related results. The results and the similarity scores (average, Functional, and Structural) were then recorded for later analysis. The goal of each sub-test was to systematically add something to the `specification` and see how the results changed.

In the first phase, we added an `object-property` pair (e.g., `Blood-Purity`) to the Initial State, beginning with a single `object-property` pair and stopping when the `initial state` of the Specification matched that of the model in the database, excluding any `value`'s.

The second phase was the same as the first, except `object-property` pairs were now added to the Objective State. Note that each phase and each sub-test are additive, so the `specification` is being built up over time.

In the third phase, we incrementally added a `value` to each `object-property` pair, creating an `object-property-value` triple (e.g., `Blood-Purity-Impure`) that matched the target model.

The fourth phase was the same as the third except that we added `value`'s to the `objective state.`

In the fifth and final phase, `structural connections` were incrementally added to the `specification` (e.g., `Renal Artery branches to Segmental Arteries`). Note that these `structural connections` were added in the order that they appear in the `behavior` of the target model so as to mimic how a user of our system might add things as he or she is thinking about the target process.

Method of the Second Test

In the second test, our goal was to evaluate our system's ability to retrieve designs of biological systems based on the problem `specification` illustrated in Figure 3. This `specification` is based on a design project in the Fall 2009 edition of the biologically inspired design course described above. This sustainable design project had the goal of conserving home water use by recycling greywater for use in toilets. The design has contaminated home water going through an elaborate filtration process, eventually being divided into blackwater, which contains the contaminants and must be sent for treatment, and greywater, which can be reused.

Next, we deployed the same `specification`-building, five-phase experiment discussed in the first test. In this second test, we start from scratch and eventually the design `specification`, given in Figure 3, instead of assembling something that already exists in the database. For example, the first `object-property` pair added was `Water-Location`; the first `object-property-value` triple added was `Water-Location-Home`; and the first `structural connection` added was `Water Contained in Home Appliances`.

Note that there was a model already in the database called "Recycle Greywater," but it used neither the same terminology nor the same structures as the design `specification` used for this test.

Results

For the first test, our system returned the "Intestinal Villi Absorb Water and Nutrients into Blood" model as a best match for the first four phases. This model was not our target model. Despite being the best match, our system only gave this model on average a 10% similarity score. In the fifth phase where `structural connections` were included in the `specification`, the system returned "Kidney Removes Waste from Blood," which was our target model, and gave it from 68% similarity (in the beginning of phase 5) to 98% similarity (at the end of phase 5). The reason we didn't get 100% similarity is because the Structural similarity metric considers `objects` from all the behavioral states in the model and we've only included information in our `specification` from the `initial` and `objective states`.

For the second test, our system returned the "Transpiration" model as the best match for all phases, ranging from a 15% similarity score in the beginning of Phase 1 to a 28% similarity score at the end of Phase 5. Additionally, the "Osmosis" model was returned as the second best match for the first four phases and first step of the fifth phase (ranging from 14% similarity to 20% similarity), and the "Root Absorbs Water" model was returned as the second best match for the rest of the fifth phase (20-23% similarity).

7 Conclusions

Since biologically inspired design is a promising paradigm for creative and sustainable solutions, there is a race to develop computational techniques and build computational tools to support the design process. However, at present both the practice of biologically inspired design and the development of supporting computational tools are *ad hoc*. In this paper, we presented an information-processing model of biologically inspired design as well as a knowledge representation scheme for organizing knowledge of biological systems from a design perspective. We then applied a case retrieval technique that utilized redundant discrimination trees for automatic indexing and retrieval of biological cases in the DANE system.

Based on the experiments described in this paper, We can conclude two things about automated retrieval in DANE. The first is that the system works as advertised when given a model structured appropriate to the indexing and retrieval algorithms. The "Kidney Removes Waste from Blood" model, for example, was indexed correctly, was retrieved after the relevant information was added to the problem specification, and was ranked by our system as the best match with a very high similarity score. Similarly, our second test showed that relevant models could be retrieved given a realistic design problem specification. All three of cases returned as best match and second-best match were related to the movement of water through a system, and although they weren't about filtration *per se*, one could speculate on how

learning the process by which plants move water across long distances with little energy (i.e., "Transpiration") might inspire a designer to create a low-energy system to move water throughout the filtration process.

However, our tests also highlight some areas for improvement. For one, our indexing algorithms failed to index about half of the models in our database. Clearly, if an automated retrieval system is to be successful, it must also be comprehensive. Second, our indexing algorithm in the first test only retrieved the correct model when `structural connections` were used in the specification. Both of these shortcomings highlight the need for additional systemization of biological knowledge for use in automated retrieval. Because our indexing scheme was designed separately from many of the models, the model-builders did not include aspects in their models that would allow them to be indexed into discrimination networks.

In addition to improving DANE's automatic indexing and retrieval scheme, we plan to expand it to the full spectrum of the biologically inspired design process described here.. Specifically, we will enable adaptation of retrieved biologically cases to fit the input specification of engineering design problems. We are also investigating mechanisms for the discovery, abstraction, and application of design patterns to further enhance resolution of the input problem specification.

Acknowledgments. This research has benefited from many discussions with Michael Helms and other members of the Design & Intelligence Laboratory (http://www.dilab.gatech.edu/). We are also grateful to Professor Jeannette Yen, Director of Georgia Tech's Center for Biologically Inspired Design (www.cbid.gatech.edu) and coordinator of the ME/ISyE/MSE/PTFe/BIOL 4740 class for her strong support and encouragement for this work. We also thank the US National Science Foundation that has generously supported this research through an NSF CreativeIT Grant (#0855916) entitled "Computational Tools for Enhancing Creativity in Biologically Inspired Engineering Design."

References

1. Bar-Cohen, Y. (ed.): Biomimetics: Biologically Inspired Technologies. Taylor & Francis, Abington (2006)
2. Benyus, J.: Biomimicry: Innovation Inspired by Nature. William Morrow (1997)
3. Vincent, J., Mann, D.: Systematic Transfer from Biology to Engineering. Philosophical Transactions of the Royal Society of London 360, 159–173 (2002)
4. Yen, J., Weissburg, M.: Perspectives on biologically inspired design: Introduction to the collected contributions. Journal of Bioinspiration and Biomimetics 2 (2007)
5. Ashley, S.: Bumpy flying. Scalloped flippers of whales could reshape wings. Scientific American 291(2), 18–20 (2004)
6. Fish, F.: Imaginative solutions by marine organisms for drag reduction. In: Meng, J. (ed.) Procs. International Symposium on Seawater Drag Reduction, pp. 443–450. Office of Naval Research, Newport (1998)
7. Fish, F., Battle, J.: Hydrodynamic design of the humpback whale flipper. Journal of Morphology 225, 51–60 (1995)
8. Reisbeck, C.K., Schank, R.C.: Inside Case-Based Reasoning. Lawrence Erlbaum Associates, Hillsdale (1989)

9. Kolodner, J.: Case-Based Reasoning. Morgan Kauffman, San Francisco (1993)
10. Navinchandra, D.: Case-Based Reasoning in CYCLOPS: A Design Problem Solver. In: First DARPA Workshop on Case-Based Reasoning. Morgan Kaufman Publishers, San Francisco (1988)
11. Navinchandra, D.: Exploration and Innovation in Design. Springer, New York (1991)
12. Maher, M., Zhao, F.: Using Experiences to Plan the Synthesis of New Designs. In: Gero, J. (ed.) Expert Systems in Computer-Aided Design, pp. 349–369. North-Holland, Amsterdam (1986)
13. Zhao, F., Maher, M.: Using Analogical Reasoning to Design Buildings. Engineering with Computers 4, 107–122 (1988)
14. Huhns, M., Acosta, E.: Argo: A System for Design by Analogy. IEEE Expert 3(3), 53–68 (1988)
15. Goel, A., Chandrasekaran, B.: Integrating Case-Based and Model-Based Reasoning for Design Problem Solving. In: AAAI 1988 Workshop on Design, St. Paul, Minnesota (1988)
16. Goel, A., Chandrasekaran, B.: Case-Based Design: A Task Analysis. In: Tong, C., Sriram, D. (eds.) Artificial Intelligence Approaches to Engineering Design. Innovative Design, vol. II, pp. 165–184. Academic Press, San Diego (1992)
17. Goel, A.: Representation of Design Functions in Experience-Based Design. In: Brown, D., Waldron, M., Yoshikawa, H. (eds.) Intelligent Computer Aided Design, pp. 283–308. North-Holland, Amsterdam (1992)
18. Maher, M.L., Gomez, A.: Case-Based Reasoning in Design. IEEE Expert 12(2), 34–41 (1997)
19. Maher, M., Pu, P. (eds.): Issues and Applications of Case-Based Reasoning in Design. Lawrence Erlbaum Associates, Mahwah (1997)
20. Pearce, M., Goel, A., Kolodner, J., Zimring, C., Sentosa, L., Billington, R.: Case-based decision support: a case study in architectural design. IEEE Expert 7(5), 14–20 (1992)
21. Sycara, K., Guttal, R., Koning, J., Narasimhan, S., Navinchandra, D.: CADET: a case-based synthesis tool for engineering design. International Journal of Expert Systems 4(2), 157–188 (1991)
22. Gebhardt, F., Voß, A., Gräther, W., Schmidt-Belz, B.: Reasoning with Complex Cases. Kluwer, Norwell (1997)
23. Börner, K., Coulon, C., Pippig, G., Tammer, E.C.: Structural similarity and adaptation. In: Smith, I., Faltings, B. (eds.) EWCBR 1996. LNCS, vol. 1168. Springer, Heidelberg (1996)
24. Smith, I., Lottaz, C., Faltings, I.: Spatial Composition Using Cases: IDIOM. In: Aamodt, A., Veloso, M.M. (eds.) ICCBR 1995. LNCS, vol. 1010, pp. 88–97. Springer, Heidelberg (1995)
25. Hua, K., Faltings, B., Smith, I.: CADRE: Case-Based Geometric Design. Artificial Intelligence in Engineering 10(2), 171–183 (1996)
26. Louis, S.: Working from Blueprints: Evolutionary Learning in Design. Artificial Intelligence in Engineering 11(3), 335–341 (1997)
27. Gómez de Silva Garza, A., Maher, M.L.: An evolutionary approach to case adaption. In: Third International Conference on Case-Based Reasoning, pp. 162–172. Springer, Berlin (1999)
28. Maher, M.L., Balachandran, B., Zhang, D.M.: Case-based Reasoning in Design. Lawrence Erlbaum Associates, Hillsdale (1995)
29. Smyth, B., Keane, M.: Design à la Déjà Vu: reducing the adaptation overhead. In: Leake, D.B. (ed.) Case-Based Reasoning: Experiences, Lessons and Future Directions, pp. 151–166. AAAI Press/The MIT Press, Menlo Park, CA (1996)

30. Pu, P., Reschberger, M.: Assembly Sequence Planning Using Case-Based Reasoning Techniques. Knowledge-Based Systems 4(3), 123–130 (1991)
31. Bhansali, S., Harandi, M.: Synthesis of UNIX Programs Using Derivational Analogy. Machine Learning 10, 7–55 (1993)
32. Barber, J., Bhatta, S., Goel, A., Jacobson, M., Pearce, M., Penberthy, L., Shankar, M., Simpson, R., Stroulia, E.: AskJef: integration of case-based and multimedia technologies for interface design support. In: Gero, J.S. (ed.) Artificial Intelligence in Design 1992, pp. 457–474. Kluwer, Dordrecht (1992)
33. Goel, A., Craw, S.: Design, Innovation and Case-Based Reasoning. Knowledge Engineering Review 20(3), 271–276 (2005)
34. Gross, M., Do, E.: Drawing on the Back of an Envelope: a framework for interacting with application programs by freehand drawing. Computers & Graphics 24(6), 835–849 (2000)
35. Yaner, P., Goel, A.: Visual Analogy: Viewing Retrieval and Mapping as Constraint Satisfaction. Journal of Applied Intelligence 25(1), 91–105 (2006)
36. Davies, J., Goel, A., Nersessian, N.: A Computational Model of Visual Analogies in Design. Journal of Cognitive Systems Research, Special Issue on Analogies - Integrating Cognitive Abilities 10, 204–215 (2009)
37. Vattam, S., Helms, M., Goel, A.: Biologically Inspired Design: A Macrocognitive Account. In: 2010 ASME IDETC/CIE Conference on Design Theory and Methods, Montreal, Canada (August 2010)
38. Fisher, Maher (eds.): Proceedings of AAAI Spring Symposium on AI and Sustainable Design. Stanford University, Stanford (March 2011)
39. Biomimicry Institute. Ask Nature – The Biomimicty Design Portal, http://www.asknature.org/
40. Chakrabarti, A., Sarkar, P., Leelavathamma, B., Nataraju, B.: A functional representation for aiding biomimetic and artificial inspiration of new ideas. Artificial Intelligence for Engineering Design, Analysis and Manufacturing 19, 113–132 (2005)
41. Vattam, S., Wiltgen, B., Helms, M., Goel, A., Yen, J.: DANE: Fostering Creativity in and through Biologically Inspired Design. In: First International Conference on Design Creativity, Kobe, Japan (November 2010)
42. Goel, A., Rugaber, S., Vattam, S.: Structure, Behavior & Function of Complex Systems: The Structure-Behavior-Function Modeling Language. AI for Engineering Design, Analysis and Manufacturing 23, 23–35 (2009)
43. Thagard, P., Holyoak, K.J., Nelson, G., Gochfeld, D.: Analog retrieval by constraint satisfaction. Artificial Intelligence 46, 259–310 (1990)
44. Forbus, K., Gentner, D., Law, K.: MAC/FAC: A model of Similarity-based Retrieval. Cognitive Science 19(2), 141–205 (1995)
45. Feigenbaum, E., Simon, H.: EPAM-like models of recognition and learning. Cognitive Science 8, 305–336 (1994)
46. Bhatta, S., Goel, A.: Model-Based Indexing and Index Learning in Engineering Design. International Journal of Engineering Applications of Artificial Intelligence 9(6), 601–610 (1996)
47. Bhatta, S., Goel, A.: Learning Generic Mechanisms for Innovative Strategies in Adaptive Design. Journal of Learning Sciences 6(4), 367–396 (1997)
48. Tversky, A.: Features of Similarity. Psychological Review 84(4), 327–352 (1997)