A Generic Scheme for Graph Topology Optimization

Matthew I. Campbell

University of Texas at Austin Department of Mechanical Engineering 1 University Station, C2200 Austin, TX 78712-0292, USA mc1@mail.utexas.edu

Abstract

This research offers a fundamental new approach to topology optimization. To date, topological synthesis approaches are simply augmentations of existing stochastic optimization techniques. While these algorithms are useful in complex parametric design spaces, they are not designed for all topology problems. This paper describes a generalized scheme for solving topological design problems. The approach combines aspects of existing optimization techniques, graph theory, mathematical programming, artificial intelligence, and shape and graph grammars. Specifically, the approach has advantages over modified parametric optimization algorithms in that 1) a two stage technique to developing solutions and searching the parametric design space more closely models the design process performed by humans and more thoroughly maps how various topologies are represented, 2) a graph grammar for generating candidate solutions minimizes the wasted search of infeasible topologies and 3) a new stochastic guidance strategy intelligently distinguishes and independently controls the variables that alter topologies and the variables that represent dimensions and parameters. This automated design synthesis method is currently being developed to synthesize the best graph for a various applications, where the choice of nodes in the graph and how they are to be connected is determined through computational search, designing the location and connection of roadways, and designing the choice and connection of processors in a chemical plant.

Keywords: topology optimization, algorithm development, multi-disciplinary

1. Introduction

Computational tools have aided the engineering design process in drafting, analysis and, to a smaller extent, synthesis. Optimization is a specific synthesis method that is useful in finding values for dimensions and parameters in an existing detailed design where the constitutive parts of the design have been determined by engineering designers. Given current design cycle times, and increases in technologies and original equipment manufacturers, designers would further benefit from computational synthesis methods that address more ambitious conceptual design activities such as the choice of elements in a design and how they are to be connected. While such topology optimization problems are currently being studied by a number of researchers, no clear commonalties or techniques exist. Some of these research endeavors have yielded intriguing results but they are founded on modifications to existing parametric optimization algorithms. This paper introduces a fundamental approach to topology optimization that overcomes the lack of efficiency and lack of solution variability that plagues current parameter optimization approaches to topology synthesis.

In developing a generic topological optimization method, I will borrow from existing optimization methods but also extend our foundation to include graph theory, artificial intelligence, and shape and graph grammars. As is discussed below, the claim of the proposed method is that the parametric optimization approach of searching from one neighboring solution to the next is an inappropriate technique for searching for topologies as it ignores intrinsic qualities of these spaces and is inefficient at moving towards optimal solutions. In order to overcome these limitations, a tree of developing solutions is incorporated into a stochastically-guided optimization procedure to more thoroughly and efficiently capture the range of solutions that exist. In this way, the proposed technique combines concepts of both gradient-based and stochastic optimization with tree-searching methods.

Parametric optimization requires that engineers phrase the design problem as a set of unknown parameters that are to be determined for optimal performance. A more challenging problem is to determine the optimal graph topology as well as the optimal parameters for a particular design problem. For example, which components need to be assembled from a catalog of available items to solve a particular electro-mechanical function as in the body-weight scale [1] shown in Figure 1a? Or how can surface micro-machined MEMS (micro-electromechanical systems) beams be configured to produce a bi-stable or even a tri-stable mechanism [2] as shown in Figure 1b? Or how should a sheet metal component [3] be cut and folded from a larger sheet so that it meets certain spatial requirements as shown in Figure 1c? Finally, how can a function structure be determined from a conceptual blackbox description of a future product [4] as shown in Figure 1d? All of these are examples of topological engineering design problems where candidate solutions are represented by a graph as opposed to a fixed set of parameters.

The design problems introduced in Figure 1 were created using modified stochastic optimization techniques that required over 24 hours of processing time. Over the next three years, the method discussed here will be implemented and applied to these design problems. It is believed that the proposed method will find successful solutions with less processing time. The success of this research will broaden the role of automated design synthesis in engineering design by pushing such techniques into earlier conceptual phases of the design process. Clearly the goal is not to replace human designers, but rather to create a more symbiotic interaction

among designers and computers to improve current engineering design processes and the overall quality of engineering designs.

2. Background Research

A search of topological optimization literature yields almost exclusively discussions of structural optimization [5, 6]. This is due to the obvious usefulness or need for topological approaches to defining structures. In this realm of research, a structure refers to beams, trusses, or plates arranged to support applied loads which are to be optimized for stiffness and/or minimum weight. Furthermore, topology in this field refers only to geometric topology where the number of holes within a two- or three-dimensional part determines its topology. In fact, mathematicians use topologies in various ways including algebraic and differential topology [7], topological vector spaces [8], and graph topology [9]. This final concept of graph topology is where our discussion will be centered. The approach used thus far (by the author and others) is to encode the graph, which is a collection of nodes interconnected by arcs, as some descriptive fixed vector of decision variables. An optimization method is then invoked to search this space. While this approach has yielded various interesting solutions to design problems, researches have been forced to limit both the representation and evaluation of their problems to operate within the limitations of these search methods. Because these topological and parametric searches create discontinuous and highly multi-modal spaces, many default to stochastic algorithms (12] all strike a balance between exploration and exploitation. This means that they are constructed to exploit local search areas by finding local optima while exploring the larger space for other fruitful areas within the search space.

Other researchers have also attempted to solve various engineering configuration or graph topology problems in the last ten years. These include research on automatically creating circuits for both passive and active applications [13]; finding a proper ordering for manufacturing operations such as rotor construction [14] and parts machinable on a lathe [15]. In robotics, Sims [16] and White, et al. [17] have evolved animal-like robots that are configured from actuators and limbs to create walking, jumping, and swimming motions. Peysakhov and Regli [18] build Lego structures that meet given spatial constraints and Shea et al. [19] construct tensegrity structures which are composed of interconnected beams and cables. Neural networks are a natural application of automatic graph construction [20]. Even wireless internet access points have been automated through invoking a genetic programming approach to positioning and interconnecting hubs [21]. In most of these approaches, researchers favor genetic algorithms because of



Figure 1: a) a configuration of electro-mechanical elements is a graph where components are nodes and their connections are arcs; this particular configuration is automatically generated as a solution to a body-weight scale, b) a sheet metal bracket is created by invoking machine operations which create a graph to solve specific spatial constraints, c) a bi-stable MEMS device is created from beam elements (nodes) connected in a particular fashion (arcs indicated which beams connect to which) to create a device with stable positions 20 µm apart, and d) the functions (nodes) of a rice cooker are determined by connecting the initial energy, material, and signal flows (arcs) of the blackbox description .

their ability to handle complex spaces that are discontinuous and have many local optima (i.e. multi-modal). Various genetic algorithm innovations have bolstered these ambitious problems, most notably variable length genotypes [22] and genetic programming [23].

The genetic algorithm approach, which reduces the representation of a genotype to a long string of ones and zeros, has benefits in that large moves can be made about the search space by making small changes to the genotype. However, in many engineering problems genetic algorithms have not worked as well as methods using appropriate integers and real numbers. Such approaches as differential evolution [24], real-valued genetic algorithms [25], extended pattern search [26], and scatter search [27] appear to find solutions more reliably and efficiently for parametric engineering design problems. This is likely to be a result of the fact that engineering design spaces are actually smooth with many useful monotonicities [28] that can be leveraged to efficiently find an optimum.

2.1. Computational Synthesis Flowchart

Designing as performed by humans is often seen as analogous to searching as performed by computers. A space of design instances is envisioned whereby each point within the space is a solution to a common design problem. Within such spaces, candidate solutions are organized such that solutions with similar configurations are close in proximity and thus through repeated modification, a search can transition from one state to another to search for the best solution. However, such an analogy does not capture the many decisions made by human designers in creating single candidates. Engineering artifacts result from a series of decisions progressing from initial design specifications to final prototype. It is evident that this searching through neighboring candidates is not a detailed enough model of engineering design. The computer has yet to match human abilities in representing design concepts, generating good solutions, evaluating the worth of such solutions, and using information from failed attempts to guide the creation of new designs. These four activities are the distinguishing traits of the experienced engineering designer, and are necessary in some extent for computational synthesis. The flowchart shown in Figure 2 highlights the four main computational design challenges of *representation, generation, evaluation,* and *guidance,* and is an effective generalization of numerous computational design synthesis methods such as traditional optimization [29] and tree searching [30].

Initially, setting up a problem involves declaring constraints and constructing objective functions, as depicted at the top of the flowchart in Figure 2. From a traditional optimization standpoint, this involves casting the problem in the negative null form. However, in more ambitious problems, the description of the problem might not clearly fit this division. The *representation* is formulated by the programmer of the computational design method to capture the forms or attributes of the design space. For example, in genetic algorithms, the representation is usually a bit-string that represents the key decision variables in the process. Using this representation, candidate solutions are *generated* in the generation task. In genetic algorithms, this is done by mutating and "crossing over" existing or parent candidates. Each generated candidate is *evaluated* in the evaluation task to determine how well it meets the objectives and constraints of the design problem. Based on the objectives calculated for the candidates a *guidance* strategy is implemented to inform the search process of how to find better solutions in the subsequent iterations. In genetic algorithms, this is the "survival of the fittest" tournament selection where candidates with inferior fitness values are removed from the search process. This example shows how the flowchart can be mapped into genetic algorithms, and it is believed that nearly every computational design synthesis method can be mapped into these four parts.



Figure 2: The generic flowchart for Computational synthesis has four basic divisions: a representation of the design space, a method for generating new solutions, a method for evaluating solutions, and a method for guiding the search process.

This flowchart is discussed in depth in a number of publications by the author [31, 32, 33], and this division can be useful for three distinct purposes. Firstly, the division of representation, generation, evaluation, and guidance provides a clear framework for researchers presenting computational synthesis or automated design accomplishments. Secondly, the division is useful in tackling new problems. A series of questions presented in Campbell and Rai [32] prompts researchers with questions to consider in approaching new problems. Thirdly, the division is useful in teaching this field of study to students new to the idea of computational design synthesis. The next four sections of the paper will discuss the fundamental representation, generation, evaluation, and guidance methods of this new generalized approach.

3. Representation

As stated earlier, the first assertion about representation is that the solutions will be represented by a computational graph. Given the obvious power and popularity of object-oriented programming [34], nodes and arcs are represented by computational structures that contain the connectivity information of the graph and various parameters for the design variables that define the specifics of the topology. For a given problem, one will need to develop a set of rules for building designs starting from an initial seed or design specification. In Figure 3, a tree of developing solution is superimposed over the design space. At the top of the tree is the problem as specified by the user. A candidate solution is constructed by committing to design decisions as represented by each branch in the tree. In this way, the proposed technique more closely mimics the design process as presented in various design theory texts (such as Pahl and Beitz [35] and Otto and Wood [36]) of moving from specifications to completed designs. Note that our topological graphs are now being stored within each node of the search tree. While a tree is also a form of a graph, each node of the tree, hereafter called states, contains each candidate topology. The proposed methodology thus extends the computational design process to include a tree of developing individual solutions. This is in contrast with traditional approaches which simply start with an existing solution and perform modifications to arrive at neighboring completed solution. One finds that it is not easy to define a "neighborhood function" for topology problems that allows a search access to the complete space of possible solutions.

The simplest approach to implementing this tree may be to view the transitions between states as adding either a single node or a single arc to the previous graph. However, this is likely too general and hence a set of rules may be constructed to prevent creating nonsensical topologies. To accomplish this, we borrow the concepts of shape [37] and graph [38] grammars. Originally the product of architecture, the grammar concept, which is also similar to production rules in cognitive psychology [39], has gained recognition in engineering design [40]. Grammars essentially provide the hard constraints in the search, thereby preventing any exploration into search space regions which the grammar rules do not allow. Past approaches to topology optimization simply penalize infeasible designs in the evaluation phase of the search process as opposed to using hard constraints to reduce the search space. These soft-constraint penalties are undesirable in topology optimization problems since search is often wasted on infeasible topologies and the introduction of penalty functions complicates the objective function.

In the sheet metal topology problem (Figure 1c) stated earlier, for example, grammar rules prevent cuts starting from inside a sheet and constrain (as a hard constraint) the size of sheet metal patches after an operation is performed. Figure 4 shows a grammar rule used in the sheet metal design research [3, 41]. The left side of the rule represents what must be present in the existing state for the rule to apply. In this case, the state must have at least two nodes where one of the two nodes is a corner node that is represented as having two adjacent edges of the sheet. If this condition is met and this 'notching' rule is applied then the resulting state will replace the original two nodes with three nodes defined by the configuration on the right. Note, that in addition to invoking this change in the graph, the resulting state is also specified by a new parameter, h, which is restricted to be no larger than H. In Figure 5, the beginning of the search tree is shown. At the top of the tree, a blank sheet represents the initial starting point in the design process. Various grammar rules deemed applicable at this stage provide the valid transformations to this initial sheet including the rule shown in Figure 4. Furthermore, each topological change also introduces dimensional parameters that need to be determined to fully describe the topology at each decision state. By carefully constructing these grammar rules, the space of feasible sheet metal components is successfully represented.



Figure 3: For purposes of topological design synthesis, it is useful to envision a search tree super-imposed above the search space. Rules are constructed to guide the generation of feasible solutions.



Figure 4: An example graph grammar rule from sheet metal design. The initial two nodes are transformed into three as a result of a corner-notching operation. Several dimensional parameters result that are bounded by the initial size of the sheet metal patches.



Figure 5: The search tree for the sheet metal design problem starts with a series of transformations that can be applied to a sheet metal blank. Rules such as corner notching, side notch, and slit all lead to valid sheet metal topologies and contain dimensions such as notch-height, position and depth.

4. Generation

A completed design solution is the result of following a series of decisions in the preceding search tree representation to make a complete topology. A 'recipe' can be created for a design by storing each decision made at each branch in the tree. Figure 6 illustrates a linked list of arbitrary length to capture the construction of the topology. There are two distinct types of decisions in this list: values which indicate the rules applied to get to subsequent states (called topology-change integers), and values chosen for parameters within each rule application. This provides us with a list structure which is easier to directly manipulate than the topology graph. An analogy can be made (as is often done in genetic algorithms) to a phenotype-genotype distinction. The actual organism or topological solution is a phenotype; it is encoded as DNA, or as a linked list in this proposed method. One might argue that topological changes and parameters do not need to be constructed with arbitrary lengths and angles. Thus, a more generalized formulation is conceived where topological changes and parameter values are intertwined. Later we will discuss what can be done if the two can be successfully decoupled.

The linked-list recipe stores values for creating a particular design, as well as the limits imposed on the range of possible values. This allows the search to modify the design afterwards without pushing it into an infeasible region. Depending on the grammar rules chosen to create the graph, linked-list recipes of different candidates may have different combinations of topology-change integers and parameters, along with potentially different lengths to their lists. Based on this linked-list recipe, one can make any number of designs by simply choosing values sequentially within the ranges of each variable until a completed candidate solution is created. In Section 6, approaches are discussed in which this generation process is guided to find the most successful or near-optimal solutions.

5. Evaluation

It is hard to imagine how one would evaluate the worth of a candidate topology in engineering without invoking some detailed simulation. Techniques in traditional optimization fall short in solving ambitious problems since they require a single objective function defined at the onset of the process. Recently, a number of fields have addressed this shortcoming in traditional optimization – specifically in aeronautics where optimization is combined with computational fluid dynamics [43], and in structural design [44]. While results have been promising, there are some common difficulties to these approaches. First, since search processes often rely on testing thousands of candidates, the time required to analyze each must be kept to a minimum. For example, a finite element simulation that takes approximately an hour would cause the search process (which must call thousands of such analyses) to take thousands of hours. Second, in order to perform a simulation, detailed pre-processing must be done for each simulation. This involves setting up the proper boundary conditions, and the resolution of the discretization (e.g. meshing of elements, or time steps) which is often performed by an engineer with sufficient experience and knowledge of the how the analysis is performed. Third, the post-processing of the simulation data is also a complicated manner. Here, the difficultly is in finding a single metric of design worth amids the output files. While the output is often presented to the user as graphs or colored images, a single quantitative measure of the worth is all that is desired for computational design synthesis. Fourth, simulations sometimes fail due to problems in calculations, such as the singularity of a matrix or unbound iteration. These analysis mishaps can cause the entire search process to halt

prematurely if not handled properly. Since this interruption results in an error in the evaluation portion of the search routine, the rest of the process becomes corrupted by erroneous or stalled analyses. Finally, engineering design problems often contain more than one measure of design worth. This leads to the problem of balancing multiple objectives. Multi-objective optimization is not a new challenge, but one that continues to be a significant hurdle in computational design synthesis.

These challenges seem to motivate research in merging existing computational synthesis methods with existing computational analysis methods. Perhaps some intermediate module could be developed to handle invoking and interpreting simulations required for evaluating candidate solutions. In the method presented here, as is often the case for optimization methods, it is assumed that a black box function or simulator has been developed to measure a design's worth. In past topological design synthesis methods [45, 16], significant development has been required to create a robust evaluator that negotiates the aforementioned difficulties. One important aspect that should be gleaned from this discussion of evaluation is that a search technique should strive to minimize the number of evaluation calls. Recent computational speed and memory advances have made this goal less relevant. However, the scope of problems addressed by topology optimization makes it significant again.

6. Guidance

The previous three sections have discussed the representation, generation, and evaluation aspects of Figure 2. While this provides the foundations for solving any generalized graph topology problem the real challenge in searching the space of solutions is addressed by the final guidance method. The guidance methods proposed in this section are conceived to appeal to the unique balance of exploitation and exploration needed for topological engineering problems. One way to understand this new domain is to consider the topology-change integers discussed in Section 4 that govern changes in the topology separate from the parametric variables in the linked-list recipe (Figure 6). If the parametric decision variables can be decoupled from the topology, that is, if subsequent topological changes do not depend on previous parameters, then one can run a final separate parametric optimization for each resulting topology. Given that resulting parametric spaces are often smooth and monotonic in engineering design (for example, objectives like weight, settling time or efficiency are rarely discontinuous functions of variables such as lengths, diameters, or values of resistance), it behooves us to perform an aggressive search for the best parameters within each topology. This is also the approach endorsed by models of the engineering design process [35] where conceptual and embodiment design involves the selection of components followed by detailed design which involves assigning values to the parameters of the artifact. Problems where parameters and topology decisions cannot be decoupled are much more difficult, requiring us to undo the graph to previous developmental stages before safely modifying the parameters. Elaborating on this categorization, we can also identify problems that lack parameters or work with fixed parameters as a simplified third division of engineering applications which has in fact been explored greatly in artificial intelligence research.

Another distinction can be made based on whether or not each state in the search tree is a viable candidate solution. In the multistable device problem and in the sheet metal component design problem (Figure 1b and 1c respectively) each additional operation made a candidate design more complicated by either introducing more cuts and bends, or more beams; but such modifications do not make the design anymore complete. Similarly many shape optimization methods such as [6] also fit into this category. In contrast, the development of the weighing machine and the function structure (Figures 1a and 1d respectively) reach a clear conclusion in the tree where no additions can be made. Furthermore, in Agarwal et al [46], a coffee maker design is evaluated at each decision point before a final design is created.

These two separate distinctions (the coupling of variables in the recipe and the ability to evaluate states in the search tree) can be used to make a table of the various graph topology problems in engineering. Figure 7 shows these distinctions creating nine possible



topological search problems, each requiring perhaps separate guidance strategies. Along the columns, there are three classes: (I) problems in which the tree terminates in solutions and intermediate states can be evaluated, (II) problems in which all states in the tree are viable solutions and evaluatable, and (III) problem in which the tree terminates in solutions but no intermediate states can be evaluated. They are listed in this order to correspond with increasing search difficulties. The rows are divided in the three classes: (A) topological problems that lack parameters or have fixed parameters, (B) problems where topological changes and parametric variables cannot be decoupled, and (C) problems where topological changes and parametric variables cannot be decoupled. Past applications are shown in the cells in black type and solution methods are superimposed as clouds above the cells and written in white type.

In the first cell of the table (I.A), we see that tree searching methods [47], which are a cornerstone of artificial intelligence, are useful and robust methods. This is especially true in applications in which one can evaluate states approaching a goal such as choosing a move in chess or a decision to go left or right in mapping a journey. As parameters are introduced in cells I.B and I.C, we can use the branch and bound technique [48]. In the remaining six cells of the table, only stochastic search techniques have been used. Given the view of the design process as afforded by the previous three sections on representation, generation, and evaluation, three new approaches are briefly presented below.

6.1. Stochastic Branch-and-Bound

As one moves into columns II and III in Figure 7, less is known about how each individual decision in the search tree will affect the quality of the solution. Without clear information to guide the process, the search must rely on heuristics and trends detected in previous solutions. While the basic branch-and-bound algorithm has existed for nearly forty years [49], a stochastic approach has only recently been developed [50]. In traditional branch-and-bound, integer-only problems are solved by successively relaxing individual integers to real numbers and finding the optimal through simple linear programming techniques. This approach only exploits the search space, and does not function well with incomplete information.

Given that a generated candidate is comprised of topology-change integers and parameters, it is proposed that smaller parameter optimization problems are solved on the parameters of each candidate graph to determine which topological changes appear to lead to the most promising solutions. By including the stochastic element proposed in Norkin et al. [50], we balance the exploiting of single topologies with an exploration of many various topologies. In general, one cannot assume that the tree is small enough for every topology to be searched; therefore the stochastic element provides some assurance that we are finding the best solution within

	I. problems in which the tree terminates in solutions and intermediate states can be evaluated	II. all states in the tree are potential candidate solutions and can be evaluated	III. problems where the tree terminates in solutions and no intermediate states can be evaluated
A. problems with fixed or no parameters	 Path planning Tree-search methods (e.g. A*) 	Stochastic Branch-and- Bound	 Lego assemblies (Peysakhov and Regli) Function Structure (Sridharan and Campbell) Feature-directed search with intern parameter tuning
B. problems where topological changes and parametric variables can be decoupled	Manufacturing process plans Branch- and- Bound	 MSE mechanisms (Kollata, et al.) Passive circuit design (Koza) <i>virtual creatures</i> (Simms, Lipson) 	 A-Design (Campbell) neural networks (Michel and Biondi) rotor construction (Deshpande and Cagan) lathe parts (Brown and Cagan)
C. problems where topological changes and parametric variables cannot be decoupled	• wireless internet hubs (Hu and Goodman)	 Sheet metal (Padhye and Campbell) Scaffolding structures (Shea, et al.) 	 Automated architecture ala SEED (Flemming et al.) Coupled shape and configuration problems Multi-agent approach

Figure 7: Topological engineering design problems can be divided six categories.

the limited time constraints. Furthermore, by separately considering topology and parameters, we can take advantage of the characteristics of the search space and more effectively exploit and explore than existing stochastic methods which make no distinction among parameters and topology-change integers.

6.2. Feature-Directed Search with Internal Parameter Tuning (FDSIPT)

Similar to the previous approach, this method leverages the smooth quality these search spaces exhibit about chosen parameters and the tree-searching quality of topological changes. When no information is provided about each decision leading to a complete design, the search process must make some inferences based on the quality of the final designs. In this approach, n topologies are created in parallel (where n would perhaps be between 10 and 100). Each of these topologies is then parametrically optimized using an aggressive and efficient method such as sequential quadratic programming [51] and sorted from best to worst based on their evaluation. Using a technique such as the TODO/TABOO learning [52] commonalities in the topologies of the best designs are identified and stored in a TODO list, and commonalities in the worst designs and identified and stored in a TABOO list.

The TODO and TABOO lists act as a library of features that should be targeted or avoided in creating new designs. The process repeats by creating another set of n designs; however, this time they are stochastically guided by the TODO and TABOO lists. This is achieved by favoring topology-change integers that fulfill TODO features and avoiding those that fulfill TABOO features.

6.3. Multi-Agent Approach

In the most complex design problems, where parameters and topological changes cannot be decoupled, we are unable to systematically approach the design problem as is done in traditional tree-searching or parameter optimization methods. Another foundation that has yet to be explored in depth is to build upon the heuristics used by human designers in solving complex design problems. This guidance technique proposes to use computational agents to encapsulate the strategies used by real designers. The approach builds on previous work in multi-agent systems in design [14, 53, 54, 55], machine learning [56] and expert systems [57]. Expert systems are a popular artificial intelligence technique used to encapsulate the decision-making skills of a human expert in a knowledge-based system to solve difficult problems such as patient diagnosis [58].

Banking on the synergy that cooperating agents produce something greater than the individual parts, the proposed technique will construct agents to cooperate to create candidate solutions. After solutions are evaluated, feedback is provided to agents based on the quality of designs they produce. In subsequent iterations, agents are probabilistically favored if they have produced good results in past iterations. Furthermore, the agents can modify their strategies based on features of successful designs similar to the TODO/TABOO aspect suggested above. In this respect, this final guidance method is quite similar to the A-Design research developed by the author [31]. The multi-agent approach developed in that research was specific to electro-mechanical configuration problems (such as that shown in Figure 1a). Results from that study indicate that knowledge-driven agents working within a stochastically guided search process can efficiently find a variety of differing topologies to solve a specified design problem.

7. Conclusions

This paper offers a new fundamental view of topological optimization. The method borrows heavily from the existing optimization literature, but incorporates research from a variety of backgrounds to develop a new basis for automated synthesis that is believed to be more effective at solving topological design problems than existing modifications to stochastic optimization. The benefits of the proposed technique are as follows.

- 1. Viewing the search process in two stages with a tree of developing solutions and a search space of neighboring solutions provides both a better model of design as performed by humans and a more thorough mapping of how various topologies are represented and how they relate to one another.
- Encoding hard constraints in a graph grammar minimizes the generation of infeasible topologies which are traditionally eliminating by soft-constraints in stochastic search.
- 3. Distinguishing topology-change integers from parameters in a linked-list recipe allows for more intelligent guidance of designs by developing separate strategies for solving these two types of variables.

The described research is not only useful in engineering design, but has merit for computer and information sciences, and any field seeking to optimize a graph topology. Results from comparing these new techniques to existing approaches will indicate which problems are best suited for the proposed techniques. Furthermore, a better understanding of the search space will be provided as an alternative to the "Cartesian space of n dimensions" view favored in current optimization research. As proposed in Figure 7, a range of problems can be solved by a range of algorithmic approaches moving from a systematic search in the upper left to more heuristic driven search in the lower right. Experiments with the proposed methods will reveal whether or not this is true. The concept of computational agents, which inherit qualities of human decision-makers and interact within a virtual environment to achieve a particular goal, has been an attractive approach to design automation. Through the developed method, one can test a variety of interactions such as competing versus cooperating agents, and agents that learn new information versus agents that have prescribed knowledge.

Since the method proposes to increase the representation abilities and improve the efficiency of the search process, more ambitious design problems may be addressed. By developing and implementing generalized modules for solving search problems such as these, researchers will be able to implement their specific representation and evaluation code to work within the proposed framework. This research topic is a culmination of a variety of our past more application-driven research experiences. Accomplishing the stated research will bolster current efforts on specific engineering design synthesis projects and provide a foundation for future research endeavors as well as provide a fundamental basis that will be useful to other researchers.

8. Acknowledgments

The authors would like to thank the National Science Foundation for supporting this work under grant award number DMI-044880.

9. References

- 1. Campbell, M., J. Cagan and K. Kotovsky, 2000, "Agent-based Synthesis of Electro-Mechanical Design Configurations," Journal of Mechanical Design, Vol. 122, No. 1, pp. 61-69.
- 2. Kollata, E., King, C., Campbell, M. I., 2004, "Design Synthesis of Multistable Compliant Structures," 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, New York, Aug 30 Sept. 1, AIAA-2004-4415.
- Padhye, S., Campbell, M. I., 2004, "Evaluation Method for the Topological Synthesis of Sheet Metal Components," Proceedings of the 2004 ASME Design Engineering Technical Conferences: Design Automation Conference, DETC2004-57254.
- 4. Sridharan, P., Campbell, M. I., 2004, "A Grammar for Function Structures," Proceedings of the 2004 ASME Design Engineering Technical Conferences: Design Theory and Methodology Conference, DETC2004-57130.
- 5. Rozvany, G. I. N., 1997, Topology Optimization In Structural Mechanics, Springer, New York.
- 6. Bendsoe M. P., and Kikuchi, N., 1988, "Generating Optimal Topologies in Structural Design using a Homogenization Method", Computer Methods in Applied Mechanics and Engineering, 71, pp. 197-224.
- 7. Borisovich, Y. G., Bliznyakov, N. M., Fomenko, T. N., Izrailevich, Y. A., 1995, Introduction To Differential And Algebraic Topology, Kluwer Academic, Boston, Mass.
- 8. Schaefer, H. H., 1999, Topological Vector Spaces, 2nd ed. Springer, New York.
- 9. Gross, Jonathan L., and Tucker, T. W., 1987, Topological Graph Theory, Wiley-Interscience, New York.
- 10. Glover, F., 1989, "Tabu Search-Part 1," ORSA Journal on Computing, Vol. 1, No. 3, pp. 190-206.
- 11. Kirkpatrick, S., C. D. Gelatt Jr., and M. P. Vecchi, 1983, "Optimization by Simulated Annealing," Science, Vol. 220, pp. 671-679.
- 12. Goldberg, D. E., 1989, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison Wesley, Reading, MA.
- Koza, J. R., Keane M.A., Bennett F. H., Yu J., and Mydlowec W., and Stiffelman O., 1999, "Automatic Creation Of Both The Topology And Parameters For A Robust Controller By Means Of Genetic Programming," Proceedings of the IEEE International Symposium on Intelligent Control, Intelligent Systems, and Semiotics, pp. 344-352.
- Deshpande, S., and J. Cagan, 2001, "An Agent Based Optimization Approach to Manufacturing Process Planning," Proceedings of the 2001 ASME Design Engineering Technical Conferences: Design Automation Conference, DETC2001/DAc-21032, September 9-12, Pittsburgh, PA.
- Brown, K.N., and J. Cagan, 1997, "Optimized Process Planning by Generative Simulated Annealing", Artificial Intelligence in Engineering Design, Analysis and Manufacturing, Vol. 11, pp.219-235.
- Sims, K., 1994, "Evolving 3D Morphology and Behavior by Competition," Artificial Life IV Proceedings, eds. Brooks & Maes, MIT Press, pp.28-39.
- 17. White P. J., Kopanski K., Lipson H., 2004, "Stochastic Self-Reconfigurable Cellular Robotics", IEEE International Conference on Robotics and Automation (ICRA04), , pp. 2888-2893
- Peysakhov, M., Regli, W. C., 2003, "Using Assembly Representations to Enable Evolutionary Design of Lego Structures," Artificial Intelligence for Engineering Design, Analysis and Manufacturing, Vol. 17, no. 2, pp. 155-68.
- 19. Shea, K., Fest, E. and Smith, I.F.C., 2002, "Developing Intelligent Tensegrity Structures With Stochastic Search," Advanced Engineering Informatics, Vol 16, No 1, 2002, pp 21-40.
- 20. Michel, O., and Biondi, J., 1995, "From the chromosome to the neural network," Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA'95). Springer-Verlag, New York.
- 21. Hu, J., Goodman, E., 2004, "Wireless Access Point Configuration by Genetic Programming," Proc. IEEE Congress on Evolutionary Computation.
- 22. Harvey, I., 1992, "Species adaptation genetic algorithms: The basis for a continuing SAGA," Proceedings of the First European Conference on Artificial Life, MIT Press/Bradford Books, Cambridge, MA, pp. 346-354.
- Koza, J. R., 1992, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, MA.
- Storn, R. and Price, K., 1995, "Differential Evolution A Simple And Efficient Adaptive Scheme For Global Optimization Over Continuous Spaces," Technical Report TR-95-012, ICSI.
- 25. Michalewicz, Z. 1994. Genetic Algorithms + Data Structure = Evolution Programs. AI Series. Springer-Verlag, New York.
- Yin, S. and Cagan, J., 2000, "Exploring the Effectiveness of Various Patterns in an Extended Pattern Search Layout Algorithm", Proceedings of the 2000 ASME Design Engineering Technical Conferences: Design Automation Conference, DETC2000/DAC-14254, September 10-14, Baltimore, MD.
- Lasdon, L., Ugray, Z., Plummer, J., Glover, F., Kelly, J., Marti, R., 2002, "Multistart Scatter Search Heuristic for Smooth NLP and MINLP Problems," Proceeding of the Computing in Economics and Finance Conference, Paper No.195.
- 28. Papalambros, P., and Wilde, D., 2000, Principles of Optimal Design: Modeling and Computation, 2nd ed., Cambridge Univ. Press, New York.
- 29. Vanderplaats, G. N., 1984, Numerical Optimization Techniques For Engineering Design : With Applications. McGraw-Hill, New York.

- 30. Russell, S. and P. Norvig, 1995, Artificial Intelligence: A Modern Approach, Prentice-Hall, Inc., Englewood Cliffs, NJ.
- 31. Campbell, M., 2000, The A-Design Invention Machine: A Means of Automating and Investigating Conceptual Design., PhD Dissertation., Department of Mechanical Engineering, Carnegie Mellon University.
- 32. Campbell, M. I., Rai, R., 2003, "A Generalization of Computational Synthesis Methods in Engineering Design," AAAI Spring Symposium Series, Stanford, CA, March 26-28.
- 33. Cagan, J., Campbell, M. I., Finger, S., Tomiyama, T., 2005, "Computational Design Synthesis: A Position Paper," submitted to the Journal of Computer and Information Sciences.
- 34. Schach, S. R., 2001, Object-Oriented and Classical Software Engineering, McGraw-Hill, 5th edition.
- 35. Pahl, G. and W. Beitz, 1988, Engineering Design A Systematic Approach, Springer-Verlag, New York.
- 36. Otto K, Wood K., 2001, Product Design: Techniques in Reverse Engineering and New Product Development. Prentice Hall, Upper Saddle River, NJ
- Stiny G, 1980, "Introduction to shape and shape grammars" Environment and Planning B: Planning and Design, Vol. 7, pp. 343-351.
- Fu, Z., De Pennington, A., and Saia, A., 1993, "A Graph Grammar Approach to Feature Representation and Transformation", International Journal of Computer Integrated Manufacturing, Vol. 6, No. 102, pp. 137-151.
- Klahr D, Langley P, and Neches R (eds), 1987, Production System Models of Learning and Development. MIT Press, Cambridge, MA
- 40. Cagan, J., 2001, "Engineering Shape Grammars," Formal Engineering Design Synthesis, Antonsson, E. K., and J. Cagan, eds., Cambridge University Press.
- Soman, A., Padhye, S., Campbell, M. I., 2003, "Towards An Automated Approach To The Design Of Sheet Metal Components," Artificial Intelligence For Engineering Design, Analysis And Manufacturing, Vol. 17, No. 4, 2003.
- Kewu L., Woodbury R., Radford, A., 1998, "A Comparison of Shape Grammars, Parametric Shape Grammars and Shape Schema Grammars," Mathematics and Design 98 Proceedings of the Second International Conference, San Sebastian, Spain 1-4 June, pp27-34.
- 43. Leitmann, G., 1962, Optimization Techniques, With Applications To Aerospace Systems. New York, Academic Press.
- 44. Kirsch, U., 1981, Optimum Structural Design : Concepts, Methods, And Applications. New York, McGraw-Hill.
- 45. Campbell, M. I., 2001, "An Automated Approach to Generating Novel MEMS Accelerometer Configurations," TEXMEMSIII: Texas-Area MEMS Workshop, Richardson, TX, June, 7.
- Agarwal, M., J. Cagan, and K. Constantine, 1999, "Influencing Generative Design through Continuous Evaluation: Associating Costs with the CoffeeMaker Shape Grammar", AIEDAM, Vol. 13, pp. 253-275.
- 47. Dechter, R. and Pearl, J., 1985, "Generalized Best-First Search Strategies And The Optimality of A*," Journal of the Association for Computing Machinery, 32(3):505-536.
- Kumar, V. and Kanal, L. N., 1983, "A General Branch And Bound Formulation For Understanding And Synthesizing And/Or Tree Search Procedures," Artificial Intelligence, 21:179-198.
- 49. Lawler, E. L. and Wood, D. E., 1966, "Branch-and-bound Methods: A Survey," Operations Research, 14(4):699-719.
- Norkin, V. I., Pflug, G., and Ruszczynski, A., 1998, "A Branch and Bound Method for Stocashtic Global Optimization," Mathematical Programming, vol. 83, pp. 425-450.
- Lawrence C.T., and Tits, A.L., 2001, "A Computationally Efficient Feasible Sequential Quadratic Programming Algorithm," SIAM J. Optimization, Vol. 11, No. 4, pp. 1092-1118.
- Campbell, M., J. Cagan and K. Kotovsky, 2003, "The A-Design Approach to Managing Automated Design Synthesis," Research in Engineering Design, Volume 14, Number 1. pp. 12-24.
- 53. Campbell, M., J. Cagan and K. Kotovsky, 1999, "A-Design: An Agent-Based Approach to Conceptual Design in a Dynamic Environment," Research in Engineering Design, Vol. 11, No. 3, pp. 172-192.
- Brown, D.C., Lander, S.E., Petrie, C.J., 1996, "The Application of Multi-agent Systems to Concurrent Engineering", Concurrent Engineering: Research and Applications, Technomic Publishing Inc., Vol. 4, No. 1, pp. 2-5.
- Miyashita, T., Yamakawa, H., 2001, "Optimum Design Using Search Agents," Proceedings of ASME 2001 Design Engineering Technical Conferences, DETC2001/DTM-21011, Pittsburgh, PA.
- 56. Mitchell, T. M., 1997, Machine Learning, McGraw-Hill, New York.
- 57. Jackson, P., 1986, Introduction to Expert Systems. Addison-Wesley, Reading. Massachusetts. Kewu L, R. Woodbury, and A. Radford, 1998, "A Comparison of Shape Grammars, Parametric Shape Grammars and Shape Schema Grammars" Mathematics & Design 98 Proceedings of the Second International Conference, San Sebastian, Spain June 1-4 pp27-34.
- 58. Hudson, D., Banda, P.W., Cohen, M.E., and Blois. M.S., 1992, "Medical Diagnosis And Treatment Plans Derived From A Hybrid Expert System," Hybrid Architectures for Intelligent Systems, pp 330-244, CRC Press.