Proceedings of the ASME 2018 International Design Engineering Technical Conferences & Computers in Engineering Conferences
IDETC/CIE 2017
August 26-29, 2018, Quebec City, Quebec, Canada

DETC2018-85648

A TWO-TIERED GRAMMATICAL APPROACH FOR AGENT-BASED COMPUTATIONAL DESIGN

Lucas Puentes
Department of Mechanical Engineering
The Pennsylvania State University
University Park, PA, USA
lup93@psu.edu

Christopher McComb
School of Engineering Design, Technology, and Professional Programs
The Pennsylvania State University
University Park, PA, USA
mccomb@psu.edu

Jonathan Cagan
Department of Mechanical Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
cagan@cmu.edu

ABSTRACT
Early stages of the engineering design process are vital to shaping the final design; each subsequent step builds from the initial concept. Innovation-driven engineering problems require designers to focus heavily on early-stage design generation, with constant application and evaluation of design changes. Strategies to reduce the amount of time and effort designers spend in this phase could improve the efficiency of the design process as a whole. This paper seeks to create and demonstrate a two-tiered design grammar that encodes heuristic strategies to aid in the generation of early solution concepts. Specifically, this two-tiered grammar mimics the combination of heuristic-based strategic actions and parametric modifications employed by human designers. Rules in the higher-tier are abstract and potentially applicable to multiple design problems across a number of fields. These abstract rules are translated into a series of lower-tier rule applications in a spatial design grammar, which are inherently domain-specific. This grammar is implemented within the HSAT agent-based algorithm. Agents iteratively select actions from either the higher-tier or lower-tier. This algorithm is applied to the design of wave energy converters, devices which use the motion of ocean waves to generate electrical power. Comparisons are made between designs generated using only lower-tier rules and those generated using only higher-tier rules.

1. INTRODUCTION

1.1. Heuristics
Through experience, designers often adopt their own "rules of thumb" which improve their performance and efficiency across the design process. These rules are referred to as heuristics. When applied to design, Yilmaz and Seifert [1] demonstrated through an observational study that heuristic-based strategies allow for a widespread variation of solutions to a given problem. They saw significant use of design heuristics in experts during design concept generation, which was beneficial to creativity and search for novel solutions. If such heuristics could be utilized more frequently by novices, there is theoretical potential for these novices to exhibit similar levels of creativity and exploration as those of experts. Cross [2] noted that, in a familiar domain, experts may have developed the ability to process information in larger quantities than novices. He further remarked that experts also have knowledge which helps them recognize underlying principles in a given design problem, as opposed to the focus that a novice may place on surface features. Ahmed et al. [3] saw that novice designers tended to search within a closely-related, small portion of a problem's design space, limited by their lack of heuristic strategies. They point out that this method resembles a trial-and-error approach, as novices lack the design strategies gained through experience. Novices may not necessarily be constrained in their search, but the thoughtfulness and efficiency required to make broad search effective leads them to instead search locally. This contrasts with the tendencies often observed in experts, who combine a series of design activities together and execute them all sequentially as a heuristic strategy. These combinations result in large jumps across the solution space, which leads to broader search for a high-quality solution and higher search efficiency.

The early stages of design are the most critical in determining the final results of the design process, since later steps in the process build upon the product of the earliest stages [4,5]. A number of studies have analyzed strategies and tools designed to assist with idea generation. Some focus on broad search throughout a solution space [6] and initial design selection [7,8], while others attempt to narrow the search by making design decisions based on previous solutions to the current

problem [9]. This study proposes a heuristics-inspired strategy for improved solution quality and efficiency during these early design stages, attempting to reap the benefits of how an expert designer may approach problem solving.

Eurisko, developed by Lenat [10], is a multi-domain computer program designed to solve problems by using heuristics as well as internally creating and adapting previously learned heuristics. Through the application of Eurisko to a series of task domains, Lenat was able to prove that heuristics are an effective strategy for problem solving, especially in domains in which exploration of the entirety of the solution space is infeasible or inefficient. Lenat's work with Eurisko laid the foundation for future work to hone in on methods and strategies for heuristic application with the goal of improving problem-solving efficiency.

Several studies have asked inexperienced designers to use high-level, abstract heuristics as a method for broadening the range of solutions for a given design problem. Daly et al. [11] observed the impact of heuristics on novice design generation, showing that designs lacking use of any heuristics were less developed and resembled currently-existing products while designs which incorporated heuristics through the design process were evaluated to be more creative and fully developed in terms of specific details pertaining to the design. Further, they showed that the use of heuristics by novices engendered search patterns similar to those of experts, as the application of these abstract design changes led the designers a significant distance from the localized region of the solution space. Kramer et al. [12] expanded on this work, showing the broad potential for general heuristics across a range of applications. These works suggest that the use of design heuristics can be beneficial by decreasing fixation on existing product designs, and thus empowering novices to exhibit the more efficient design behaviors of experts.

1.2. Design Agents

A design solution is typically formulated through a series of changes and evaluations. This iterative process continues until a predetermined end point has been reached, which commonly includes a specified number of iterations or timeframe, solution convergence to an optimal point, or completion of design goals. Due to the repetitive and tedious nature of such a process, the use of computational agents for design synthesis can significantly decrease the amount of time needed to generate a problem solution [13,14]. Provided with the rules to create and evaluate valid design solutions, agents have the capability to rapidly apply and assess potential solutions. Agents can be given sets of preferences to guide them throughout the design process, while a human designer can cooperate within this cycle by analyzing agent solutions and making adjustments to the agent's design preferences. Campbell et al. [13] demonstrated the viability of dynamic preference updates through A-Design, an agent-based algorithm focusing on the adaptation of agents to changes in a human designer's preferences. A-Design demonstrated that early solutions to a given design problem were of poor quality, but higher quality solutions converged to satisfy those preferences as the agents continued iterations. Moss et al. [15] later built upon A-Design, proposing that an understanding of the cognitive factors that come into effect during design have the potential to greatly improve the capabilities of automated computational design. Incorporating shape grammars into agent-based design, Orsborn and Cagan [16] presented a method for automated design generation driven by user or consumer preferences.

Other studies have explored how designers perform individually compared to within a team organization, with each team member targeting the same goal. Research suggests that interaction between team members leads to better individual performance than if they were working alone [17,18]. Many computational design algorithms are team-based in an attempt to capitalize on some of their benefits [19–21]. Each agent runs its own iterative design process. After a determined time length, the team collaborates to share solutions and to evaluate where each agent is to begin the next iteration cycle. Teams can be given specific traits in order to more accurately resemble effective human tendencies, such as biases displayed during team evaluation periods. Designers utilizing approaches such as these must be cautious though, as improper assembly and management of agent teams can be detrimental to the final design solution produced by the team [20]. For these reasons, the current study uses the Heterogeneous Simulated Annealing Team (HSAT) algorithm, as it has been shown helpful within agent-based computational design [21]. Computational agents iteratively work individually on developing solutions to the design task and then collaborate at set intervals to share solutions and evaluate from where to continue iterating. This framework is discussed further in Section 2.2.

1.3. Proposed Framework

Our proposed approach in this work is the creation of a two-tiered grammar that will enable computational design algorithms to reason using both standard grammatical representations (the lower tier) and more abstract heuristic strategies (the higher tier). The lower tier of this approach contains rules which are domain-specific and apply minor changes to the current solution under iteration by a computational agent. The higher-tier of the approach contains transformation rules which are more abstract and apply major changes to an agent's current solution, similar to the heuristic approaches used by experts. The level of abstraction of these rules allows for re-use across a variety of domains. A generalized visualization of this approach can be seen in Figure 1. The implementation of this approach within the HSAT computational design framework enables agents to intelligently search distant areas of the design space for high-performing solutions, while at the same time locally improving designs.

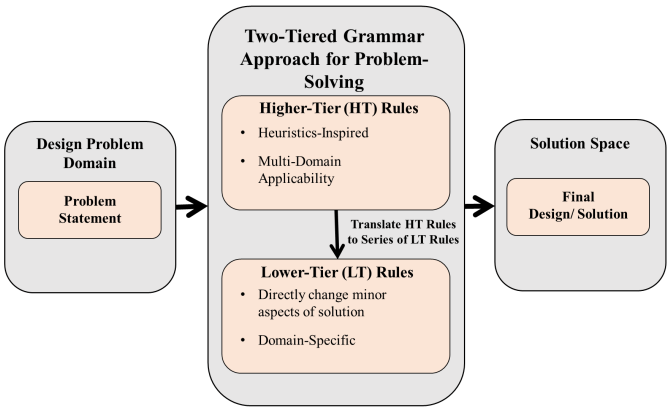


FIGURE 1. GENERALIZED IMPLEMENTATION OF TWO-TIERED GRAMMATICAL APPROACH WITHIN A DESIGN PROBLEM

The remainder of this paper is organized as follows. Section 2 reviews relevant background pertaining to graph grammars, agent-based methodology used in this work, and the wave energy design problem to which it is applied. An explanation of wave-energy converter operation and design potential is given. In Section 3, the implementation of a two-tiered grammar is described, as well as its application within design generation of wave energy converters. Section 4 presents the results of the application and findings suggested by the data. Comparisons between designs generated using only lower-tier rules and higher-tier rules are examined. Section 5 concludes this paper and details limitations and potential future work regarding two-tiered grammars.

2. BACKGROUND AND RELATED WORK

2.1. Graph Grammars

Graph grammars are a commonly used method of design representation using nodes and connections. The nodes and connections have associated parameters which denote properties of the object they have been defined to represent [22–25]. Changes are made to a design through the addition, removal, and manipulation of nodes and connections using defined rules. The goal is to generate a solution to a design problem within the domain for which the rules have been defined. Various design search strategies which have been developed to optimize design problem solutions can be applied to graph grammars. One such method, genetic algorithms, synthesizes multiple solutions in a combinatorial manner inspired by evolutionary concepts [26] and has been utilized for optimization within engineering problems [27]. Another strategy, simulated annealing, incorporates a dynamic probabilistic method of accepting new design solutions after applications of grammar rules [28]. Early work done using graph grammar-based design for optimization by Schmidt and Cagan [24] applied a simulated annealing-driven graph grammar to cart design using Meccano Set components. That study both successfully demonstrated the ability of a graph grammar to be used in design optimization and suggested that its implementation allows for complete search within a design space. Later works utilized graph grammar-based design in gearbox design synthesis [25,29], passive dynamic brachiating robots [23], and optimization of photovoltaic arrays [30], among others.

Creation of a graph grammar is a process which involves defining the vocabulary as it applies to the design task, defining a set of valid grammar rules (actions taken to modify a design), defining an initial design, and generating a design using the grammar rules [22]. The language can be modified repeatedly to allow changes in how and where rules are applied. Grammar rules are typically viewed through a "left-hand side" and "right-hand side" of each rule. The left-hand side shows the state of a design prior to application of a rule, while the right-hand side shows the results of the rule application. This left-hand side to right-hand side transformation is implemented within the full design by matching the applied rule to a sub-graph of the current design. In developing a graph grammar, Knight [31] pointed out the dilemmas a designer may face during the process. At some point, a clear connection has to be made between the grammar rules created and goals that may arise for a design problem requiring use of said grammar.

Another issue that even experienced designers may face is the unpredictability of the results of even the simplest of rules. Grammar creators need be cautious in the definition of rules, as a poorly designed grammar can be inefficient and nonproductive. Further, if a grammar is established with a limited, problem-specific target in mind, the grammar may not fit the needs of other problems within its domain. In returning to a previously designed grammar for an epicyclic gear train for a new design problem, Li et al. [32] found that the existing grammar rules were not sufficient to complete the task at hand. Expansion of the grammar rules were required in this case to allow for a broader range of generated solutions. Tools to assist in grammar creation, such as the grammar rule analysis method (GRAM) [25], have been introduced to aid designers in creation of fully-developed grammars. GRAM provides feedback to grammar developers to assist them in considering new rules, omitting unnecessary, and altering inefficient rules.

2.2. Heterogeneous Simulated Annealing Teams

As mentioned previously, computational agents can be organized into team-like structures. The Heterogeneous Simulated Annealing Teams (HSAT) framework employs a team of computational agents which engage in a locally sensitive search and quality-informed solution sharing [21]. The locally sensitive search characteristic is achieved through an agent's use of an adaptive simulated annealing algorithm in accepting design changes across individual iterations. After a design change has been applied to the current solution of an agent, this candidate solution is evaluated, and its results compared to the results of the current solution. If the solution is evaluated as an improvement, then the agent adopts this candidate solution as its new current solution. If the candidate is not an improvement to the current solution, then it is probabilistically accepted based on Eq. (1):

$$p = \exp\left(\frac{f(x_{new}) - f(x_i)}{T_i}\right) \quad (1)$$

where p is the probability that candidate solution, x_{new}, will be accepted as the replacement to current solution, x_i. The evaluation of the objective function of a solution is designated as $f(x)$, and the current temperature of the agent is T_i. Once the agent has determined whether or not to accept a new solution, the agent's temperature is updated using the Triki temperature scheduling [33]. The Triki schedule calculates the temperature of an agent given index i through Eq. (2):

$$T_{i+1} = T_i\left(1 - \frac{T_i * \delta_T}{\sigma^2_{f(x)}}\right) \quad (2)$$

where T_i is the temperature at index i, δ_T is a parameter controlling the speed of adaptation, and $\sigma^2_{f(x)}$is the variance of candidate solutions observed since the last temperature update. In cases where the variance of candidate solutions is zero, the variance is adjusted to be a very small value in order to prevent a division by zero error.

After a set number of iterations has passed, all agents collaborate to accomplish the second focused characteristic of HSAT, quality-informed solution sharing. The current solution for each agent is acquired and stored within the vector **F,** Eq. (3):

$$\mathbf{F} = [f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), \ldots, f(\boldsymbol{x}_N)] \quad (3)$$

where $f(\boldsymbol{x}_k)$ is the objective function of the current solution of agent k. In a design problem with a goal of maximizing the objective function of a solution, each solution within **F** is compared to the worst current solution in order to create the weight vector **W**, Eq. (4)**:**

$$\mathbf{W} = \mathbf{F} - \min(\mathbf{F}) \quad (4)$$

Each individual agent probabilistically selects a solution from the list of available solutions as a starting point for its next iteration according to Eq. (5):

$$j = \text{mult}\left(\frac{\mathbf{W}}{\sum_i W_i}\right) \quad (5)$$

where the "mult" function designates that the agent is selecting from an applied multinomial distribution formed by a proportional weighting of each solution within the weight vector **W**. The collaboration frequency of agent teams is selected by the designer to achieve a balance between divergence and collaboration. Higher frequencies tend to result in consistently low divergence, while lower frequencies can allow agents to diverge substantially. The tradeoff between high and low frequencies is that the higher frequencies are likely to result in a more refined design spanning across a small number of branches in a solution space, while lower frequencies are likely to span across more branches in the solution space at the cost of local refinement. Because this solution sharing is performed probabilistically rather than deterministically, agents may not necessarily select the current top solution and thus can continue making changes to alternative designs. For the design problem used in this paper, collaboration is set to occur every 10 iterations to allow agent teams to explore a common region of the solution space as they work towards a final design After this collaboration period, each agent continues to iterate through design solutions until either the next collaboration period or when the end of the design process has been reached. At this point, a final design is selected by the team. Implementing our two-tiered grammar within HSAT expands agents' search for solutions in the design space.

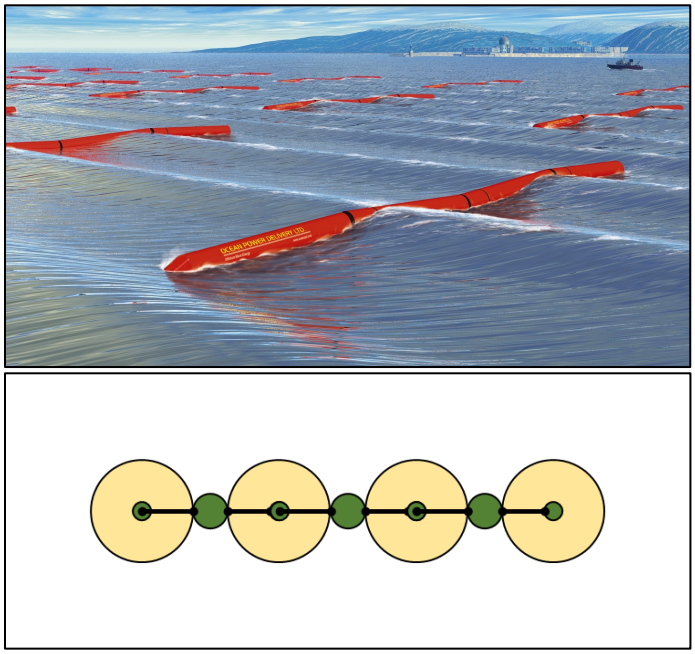
2.3. Designing Wave Energy Converters

Renewable and sustainable energy methods have long been popular areas of research within the engineering community. Wave energy converters (WECs) are a type of sustainable energy device that converts kinetic energy from ocean waves into electrical power. This is accomplished through one or more power take-off units (PTOs) inside the WEC [34,35]. Floating bodies in a wave energy converter system are connected using these power take-off units, while using the relative motion of the oscillating bodies to generate power within the unit. Most PTOs fall into two main classifications: rotational PTOs, which transform the energy from the rotational motion of the bodies about the joint into electrical energy, and linear PTOs which transform the relative translational motion between bodies into electrical energy [36].

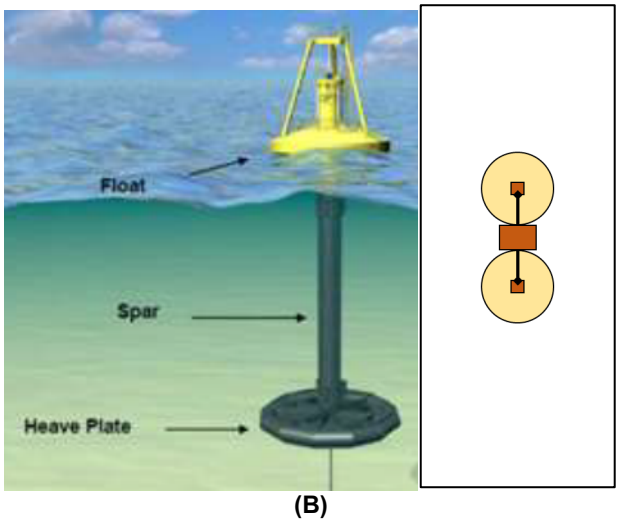
Because WECs are still an emerging technology, there has been no convergence on a standard WEC design [37]. Depending on a series of location-based factors (e.g., ocean depth, typical wave frequency), engineers must select the WEC design that will be the most effective. This means that engineers must design WECs which may be drastically unlike others in relatively similar scenarios, or even invent novel solutions which have not been used in a prior design. An example of the vast differences in viable converters can be seen in the Pelamis device compared to Ocean Power Technology's PowerBuoy device [34,37]. The Pelamis, tested off the shores of Scotland, is a snake-like chain of four cylinders which generates power based on rotational motion of the cylinders about its joints. The PowerBuoy, also tested off the coast of Scotland, has a much different method to power production, converting wave energy through the linear motion between a large circular floating disk and a submerged buoy.

This wide range of WEC configurations contrasts with other highly-developed renewable energy types. For instance, utility-scale wind energy conversion device designs have primarily converged on a three-blade horizontal axis wind turbine, with minor adjustments made to meet more specific requirements in the location they are utilized [38]. Despite the lack of standardization in WEC design when compared to other renewable energies, wave energy has been estimated to possess the highest energy density and potential active power generation time among renewable resources [37]. The problem-specific

requirements of a WEC allows great utility for application of computational agents to iteratively generate top solutions. For this reason, along with the relevance of renewable energy systems, the design of a WEC is selected as a case study for implementation of this paper's two-tiered grammar rule approach. Figure 2 demonstrates how current WEC topologies, specifically the Pelamis device and the PowerBuoy, can be portrayed through the graph grammar representation introduced in the next section. Table 1 in Section 3.1 serves as a legend to identify components shown within this grammar.



(A)



(B)

FIGURE 2. REPRESENTATION OF (A) THE PELAMIS DEVICE [39] AND (B) OCEAN POWER TECHNOLOGY'S POWERBUOY [40] WITHIN THE GRAPH GRAMMAR DEVELOPED FOR THIS STUDY

3. METHODOLOGY

In this work, we propose a novel two-tiered grammar approach for design generation. Within the domain for a given design problem, the lower-tier consists of domain-specific rules which apply minor design changes, while the higher-tier consists of domain-agnostic, heuristics-based rules which apply major changes. This approach is utilized by a team of computational agents within a design problem for wave energy converters. The impact of each individual tier on the quality of the agents' final design is examined.

3.1. Two-Tiered Grammar Rule Approach

At the core of this two-tiered grammar is the notion that higher-tier rules are to be abstract and heuristics-based, with potential for application across a variety of domains. These rules are inspired by expert designers' tendencies to make large but informed leaps across the solution space to search for high quality designs. The use of these higher-tier rules allows for quick distant searches within the solution space, and thus a variety of differing design topologies can be explored. Lower-tier rules are specific to the domain of a given design problem, and as such should be valid for any given problem in said domain. These lower-tier rules apply small changes to the graph design, assisting with local refinement. Contrasting the higher-tier rules, the lower-tier is aimed at parameter-level design, with the intention of perfecting the design topology that has already been decided upon. A higher-tier rule must be translated, directly or indirectly, into a series of lower-tier rule applications before it can be applied to the design problem. The series of lower-tier rules derived from the use of a higher-tier rule are guided by the heuristic within the higher-tier rule, as opposed to the standard design quality-driven series of lower-tier rules. Although higher-tier rules are intended to have utility across multiple domains, there may be instances where one or more higher-tier rules cannot to be translated down into lower-tier rules in some domains. Because the purpose of higher-tier rules is to implement a guided approach to applying a series of lower-tier rules, non-relevant rules may be replaced with those which are applicable. The combination of these two rule tiers is aimed at covering both a broad range of topologies as well as refined designs across the solution space of a domain.

A wave energy converter design problem is used in this work as an application for the proposed two-tiered grammar rule approach. The goal of the problem is to maximize the objective function, chosen as power density (in W/kg), of a designed WEC. WEC devices are simulated using a combined time- and frequency-domain simulation [41,42]. A set of lower-tier grammar rules is first established for the domain of WEC design. These rules allow for addition, removal, relocation, and parameter adjustment of floating bodies and PTOs. This lower-tier rule set is based on examination and exploration of existing WEC device designs. This derived grammar is deemed effective as it has the capability to recreate existing devices. Next, a set of applicable higher-tier rules is created. The rules selected for this tier are chosen as heuristics that are potentially generalizable to other domains. This higher-tier rule set is

selected as a sample of heuristics that enable topology modifications typical in configuration design. Partial inspiration for these heuristics comes from design heuristics identified by Daly et al. [11] Additional rules may be added to the grammar in either rule tier if relevant to the current design problem. Symbols used within the grammar are provided in the legend in Table 1. All of the higher-tier and lower-tier grammar rules utilized within this work are provided in Table 2 and Table 3, respectively.

3.2. Agent-Based Design Algorithm

A team of computational agents is organized according to the HSAT framework to apply the two-tier WEC grammar for the creation of a WEC design with maximized power density. The agent team performs a number of total iterations (individual rule applications/assessments) per every design repetition (the complete generation of a final design), as per [21]. Each agent is instantiated with the same preferences in rule-tier selection and method for solution evaluation. At the beginning of each agent's individual iteration, the agent probabilistically chooses which rule tier to select using its rule-tier preferences as the weighting values for this selection. For example, if preferences are set to 80% for lower-tier and 20% for higher-tier, the agent has an 80% chance to choose a rule from the lower-tier and a 20% chance to choose from the higher-tier. Preferences can further be set to definitively select from one specific tier by weighting that tier 100% and the other 0%. If desired, tier preference also can be updated at the end of each iteration to allow for different tier weighting in the subsequent iteration. For the purpose of this work, comparing the results of a design constructed by an agent team using solely one of the two rule tiers, the preferences of the agent teams are static and definitively select from only one tier. Once the tier is chosen, the agents stochastically apply a rule from that tier. If the new design is valid (i.e. bodies do not overlap), the agent decides whether to reject or accept the change in accordance with simulated annealing methodology. In this problem, the solution quality is defined as the power density (in W/kg), obtained by dividing the power of the designed WEC by its mass. Every agent independently updates their temperature every 10 iterations using the Triki scheduling in Eq. (2). The initial temperature and Triki parameter chosen for every agent were $3.0x10^{-3}$ and $6.0x10^{-3}$, respectively, based on values used previously in the HSAT algorithm [21]. Because this initial temperature and cooling rate is not specifically optimized to match the agent's rule-tier preferences, the same values are used in all preference settings.

HSAT agents also intermittently interact via quality-informed solution sharing. For this study, agents collaborate after every 10 iterations, allowing the agent to attempt multiple rule applications before considering adopting another agent's design. In instances where higher-tier rules are in use, collaboration between computational agents is of high importance because most higher-tier rules encourage divergence within the agent team. This divergent behavior aligns with observations made by Daly et al. [11], in which the implementation of a single heuristic was seen to be applicable in a multitude of ways. Once an appropriate number of iterations has occurred and the solution quality of all agents within the team has converged, a final collaboration period is used to select a final design solution.

3.3. Evaluation of Approach Based on Wave Energy Converter Design Problem

Each potential WEC design is evaluated according to its maximization of the objective function, power density, which is a function of its power output (which should be maximized) and its mass (which should be minimized). Here, mass is a proxy for cost, which has been shown to yield similar optima as direct cost minimization [43]. Solution quality produced by an agent team thus corresponds to a system which can generate the most possible power for the least possible cost. The mass of a floating body is found by multiplying its density by its weight. To calculate total system mass, the individual mass of each floating body in the WEC design is simply added together. To calculate power of the system, for every PTO in the design, its individual power based on the motion of the bodies connected to it is calculated. The type of PTO determines which of two possible power equations to use. Rotational PTOs utilize Eq. (6) to calculate instantaneous power, P (in W):

$$P = \kappa(|\omega_A - \omega_B|)^2 \tag{6}$$

where κ is the angular damping coefficient of the PTO (in N-m-s/rad) and ω is the angular velocity of bodies A and B (in rad/s). Similarly, linear PTOs utilize Eq. (7) to calculate power:

$$P = k(||v_A - v_B||)^2 \tag{7}$$

where k is the linear damping coefficient of the PTO (in N-s/m), v is the linear velocity of bodies A and B (in m/s). The power value calculated for each PTO is then summed to produce the total instantaneous system power output. Averaging this value across a simulation yields average power production.

For a single repetition of the algorithm, three computational agents are instantiated and arranged according to the HSAT organization. The use of three agents for this work is selected to allow a number of potentially divergent solutions to be explored without severely increasing computational time. The same initial design is used across all repetitions and preference distributions tested. After each agent has completed one full repetition of 500 iterations (chosen to allot simulations adequate time to demonstrate rule-tier effectiveness), the agent team collaborates a final time and conveys its final design. The process is repeated for 75 repetitions after which the objective functions of all final designs are analyzed.

TABLE 1. LEGEND FOR GRAPH GRAMMAR FOR WAVE ENERGY CONVERTER DESIGN DOMAIN

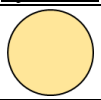
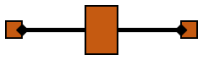
Component	Symbol
Floating body	
Rotational PTO	
Linear PTO	

TABLE 2. HIGHER-TIER GRAPH GRAMMAR RULES FOR WAVE ENERGY CONVERTER DESIGN DOMAIN

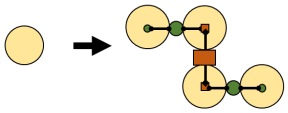
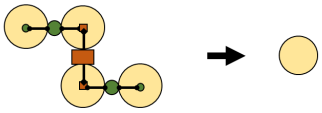
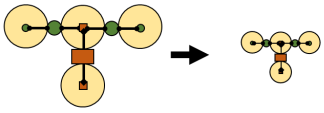
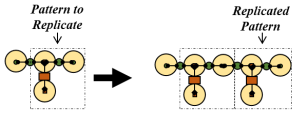
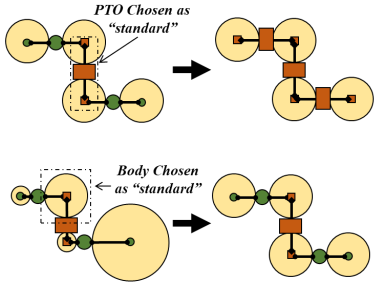
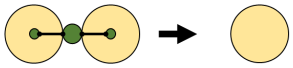
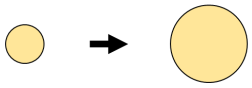
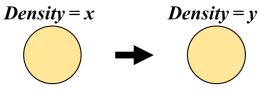
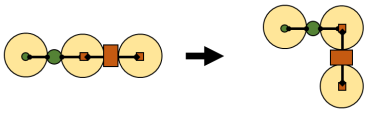
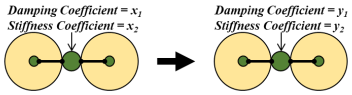
Higher-Tier Grammar Rule	Graph Transformation
H1: Increase complexity	
H2: Decrease complexity	
H3: Change design scale	
H4: Replicate pattern	
H5: Standardize	

TABLE 3. LOWER-TIER GRAPH GRAMMAR RULES FOR WAVE ENERGY CONVERTER DESIGN DOMAIN

Lower-Tier Grammar Rule	Graph Transformation
L1: Add rotational body	
L2: Add linear body	
L3: Remove body with joint	
L4: Change joint type	
L5: Change body dimensions	
L6: Change body density	
L7: Relocate body with joint	
L8: Swap bodies	
L9: Change joint coefficients	

In order to observe the efficacy of using each rule tier separately, design problem simulations are conducted using solely one tier at a time. By adjusting the tier weights designed into the computational agents, each agent's preferences are initialized to use only one of the two tiers, with all agents using the same tier. It is predicted that the use of only lower-tier grammar rules would result in local improvement of the starting design, while utilizing only higher-tier rules would result in infrequent, although potentially significant, jumps to different topologies within the solution space.

The implementation of the higher-tier rules is achieved through a heuristic-guided series of lower-tier rule applications. "Burst algorithms" have been utilized in grammar models to apply a series of rules to gearbox synthesis and bicycle frame synthesis [44]. Within each rule "burst", multiple rules are

applied to the current design, regardless of how prior rules in the burst affected solution quality; quality is assessed only after the burst is complete. To compare the effects of a guided series of lower-tier rules to an unguided series of lower-tier rules, a set of simulations is conducted in which agents apply three random lower-tier rules in a burst to the design per each iteration. Because the number of rules could be dependent on the current design (in such cases where a higher-tier rule makes changes to the entire system) and some rules are only limited by how the grammar designer implements them (for example, placing an upper limit on how much the system can expand in one higher-tier rule application), a definite determination of an average number is not feasible. A burst length of three lower-tier rules is chosen as an estimate of the typical number of lower-tier rules that a higher-tier rule translates into. The set of simulations utilizing only this lower-tier burst method is then compared to the results of using solely higher-tier rules.

4. RESULTS AND DISCUSSION

Randomly-selected samples of final WEC designs produced by agent teams utilizing only one rule tier are provided in Figure 3. Samples show final design topology along with solution quality. Surprisingly, WECs produced using lower-tier rules often are more complex than those produced from higher-tier rules. Applying only minor changes at a time, the lower-tier rules grant agents the ability to gradually build up and vary their designs throughout the creation process. The higher-tier is less likely to make changes on that same small scale, resulting in changes that attempt to expand complexity too much in one iteration. Once a higher-tier design reaches a high quality topology, large changes become less effective for improvement.

The hypothesis that designs produced using higher-tier rules become fixed on a certain topology is further supported by analyzing how solution quality changes throughout a run of the algorithm. Figure 4 shows a comparison of how solution quality improves as agent teams iterate for each rule-tier preference: lower-tier only, higher-tier only, and bursts of 3 consecutive randomly-selected lower-tier only. WEC designs generated with only higher-tier rules converge on their maximum quality much sooner than those produced by lower-tier rules. The quality of these higher-tier designs is initially higher than that of lower-tier designs, but the average solution quality of lower-tier designs catches up to and surpasses the quality of higher-tier designs after convergence. Higher-tier designs rapidly reach high solution qualities, plateauing at just above 400 W/kg before the agent teams have even hit their 100th iteration. Lower-tier designs have only improved to around half of this value once the higher-tier designs begin to converge. By the 75th iteration, higher-tier designs have nearly ceased improvements, while the lower-tier designs gradually decrease the gap between the two preferences' solution qualities until the 150th iteration. Higher-tier designs show little improvement after convergence, remaining around the same solution quality until the end of the simulation's 500 iterations. On the other hand, lower-tier designs continue to rise in solution quality and finish the simulation at a solution quality of around 600 W/kg, nearly 50% better than the higher-tier designs, and still appear to have not reached convergence yet. These final run-time solution qualities can be seen side-by side in Figure 5.

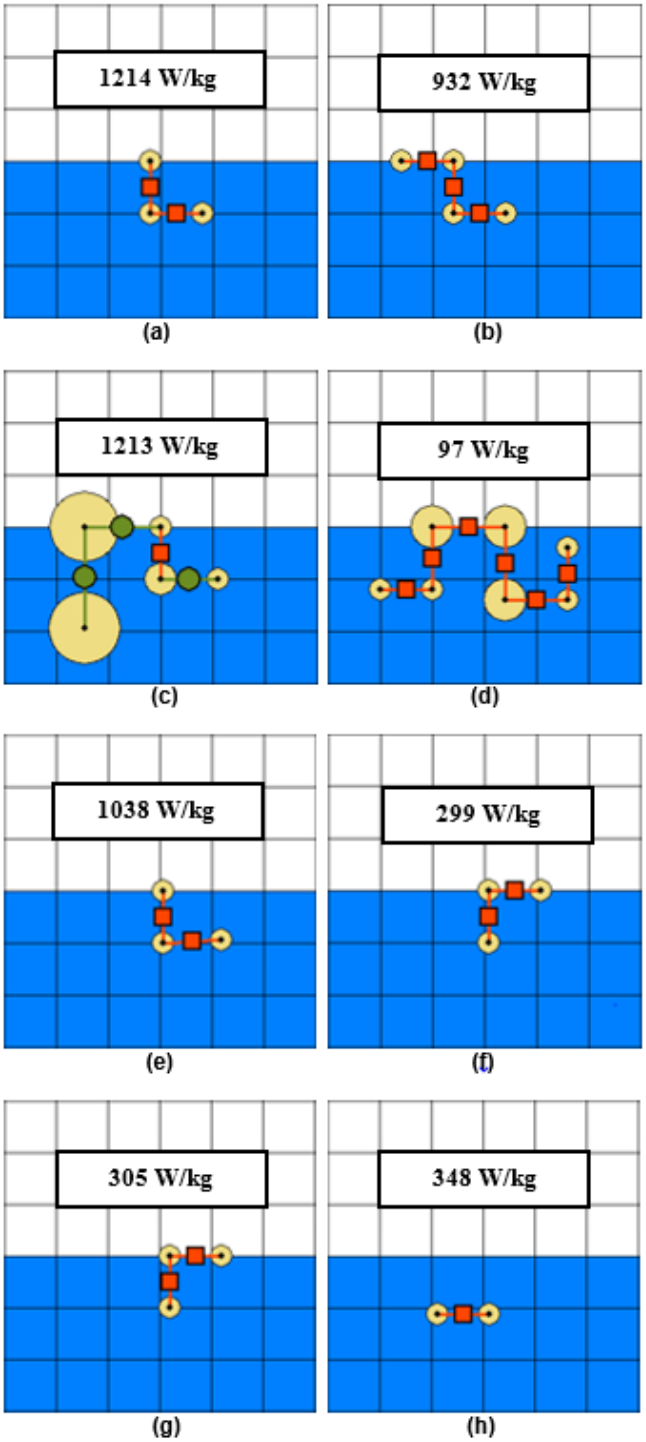


FIGURE 3. SAMPLE WEC DESIGNS GENERATED BY SOLELY LOWER-TIER RULES (a-d) AND HIGHER-TIER RULES (e-h) WITH SOLUTION QUALITIES. GREATER QUALITIES CORRESPOND TO BETTER SOLUTIONS.

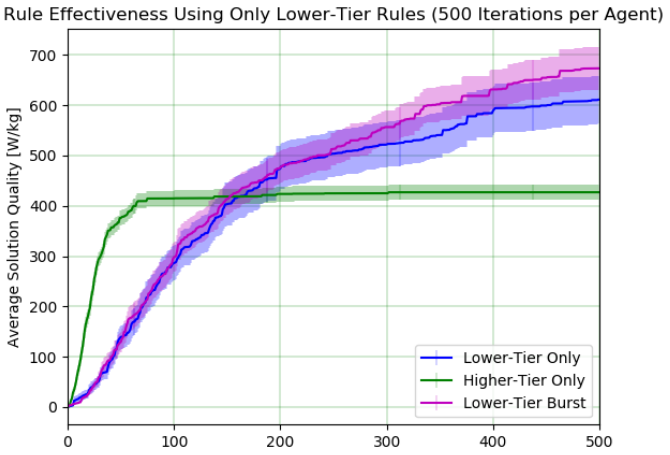


FIGURE 4. COMPARISON OF AVERAGE SOLUTION QUALITY FOR WEC DESIGN PROBLEMS OVER AGENT TEAM ITERATIONS USING SOLELY ONE RULE TIER. SHADED AREAS SHOW ±1 STANDARD ERROR.

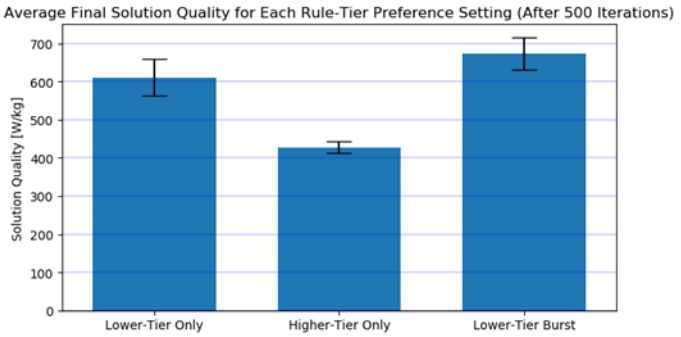


FIGURE 5. COMPARISON OF AVERAGE FINAL SOLUTION QUALITY OF WEC DESIGNS AFTER 500 ITERATIONS USING SOLELY ONE RULE TIER. ERROR BARS SHOW ±1 STANDARD ERROR.

Designs generated by randomly applying bursts of three lower-tier rules per agent iteration follow a similar trend in solution quality compared to lower-tier only designs. These designs gradually improve in solution quality as teams iterate, ultimately leading to a design quality average slightly above lower-tier designs applying a single rule per iteration. Like the other lower-tier designs, solution quality is initially much lower than that produced by higher-tier designs, while significantly surpassing them once the higher-tier designs begin to converge. This similarity of lower-tier burst designs to lower-tier single rule designs could be attributed to the fact that they both share the same unguided application of lower-tier rules which are applying small design changes with each use. However, as the number of lower-tier rules applied per burst is based on the typical number of lower-tier rules derived from each higher-tier, this demonstrates that the value of the higher-tier rules is not merely in the application of lower-tier rules but rather in applying the correct rules in the correct order, enabling an effective heuristic-guided search. Across each application, higher-tier rules are constantly based on the same driving heuristic and thus the same set of lower-tier rules, while random lower-tier bursts are inconsistent and more reliant on chance.

The series of lower-tier rules that are implemented via higher-tier rule selection results in more effective designs than the lower-tier burst approach where there is no logic for the choice of rule applications (and may even be counter-productive where a component is added and then removed). If resources are limited, higher-tier rules are the best choice to get to a good solution quickly. However, if resources are not an issue then the lower-tier rules produce far better solutions in the end.

Because the higher-tier rules increased solution quality much more rapidly than the lower-tier rules, at the very least higher-tier rules could be useful to boost traditional algorithms that primarily employ lower-tier rules. Thus, a shifting hybridization of the two rule tiers may be most effective, whereby the algorithm begins with higher-tier rules and, at some threshold of convergence, shifts to lower-tier rule application. This mixture of tiers could provide agents the ability to quickly jump to better design solutions and then fine tune the design or even explore new topologies at a more leisurely pace. This also naturally mirrors the procession of design from abstract and coarse modifications to detailed, incremental improvements.

It is important to note that the number of rules within the higher-tier is only 5, nearly half that of the 9 total rules contained in the lower-tier. Having a smaller number of rules to stochastically select from increases the likelihood that a rule which improves the quality of the design in chosen. As noted by Königseder and Shea [25] in their development of a graph grammar rule analysis method, there arise situations within a grammar in which rules may be irrelevant or not effectively defined. For some lower-tier rules contained within the WEC grammar, such as *L5: Change body dimensions* and *L6: Change body density*, efficiency of the entire lower-tier may be altered if these rules were combined into one, reducing the number of rules the agent randomly chooses from. Additionally, taking an in-depth look at rule patterns which produce positive design results can assist agents in making intelligent rule selections as opposed to stochastic ones.

5. CONCLUSIONS

This work presents a heuristics-inspired, two-tiered approach to computational design synthesis. Teams of computational agents iteratively apply graph grammar rules from one of two rule tiers. The domain-specific lower-tier rules apply minor transformations to an agent's current design. The higher-tier rules incorporate heuristics and abstract strategies in order to perform efficient and intelligent design jumps to distant points in the solution space. This proposed design methodology is implemented to design wave energy converters (WECs).

In assessing the impact of utilizing only one rule tier or the other, it is seen that heuristic rules from the higher-tier ruleset are initially more efficient than lower-tier rules in reaching a high quality design solution. Over a longer timeframe, however, higher-tier rules are unable to continue to improve upon the design, while lower-tier rules have the capability to continue

design improvement. These higher-tier rules are guided by abstract heuristic strategies, and thus search the initial design space more efficiently by minimizing the number of solution evaluations that are necessary. Applying bursts of unguided lower-tier rules per each design iteration did not have the same efficiency as the guided lower-tier rules determined by the higher-tier, though did eventually surpass the higher-tier approach. Further exploration on the benefits of this two-tiered grammar approach can be accomplished through initializing the WEC design problem with alternative designs and through application to design problems in other domains.

To capture the effects of a combination of both rule tiers, future work will employ simulations with a shifting hybridization of the two rule tiers to identify effective hybrid patterns. The mixture of rule tiers could provide agents with the ability to quickly jump to high quality solutions and then fine tune those solutions from that point, or attempt to perform large jumps again once solution quality begins to converge. As well, using tools to analyze the grammar rules themselves opens up the possibility of a more refined two-tiered grammar that will empower agents to more effectively use of the rules contained in the grammar.

ACKNOWLEDGMENTS

This material is based upon work supported by the Defense Advanced Research Projects Agency through cooperative agreement N66001-17-1-4064 and the Air Force Office of Scientific Research through grant number FA9550-18-1-0088. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the sponsors.

REFERENCES

[1] Yilmaz, S., and Seifert, C. M., 2011, "Creativity through Design Heuristics: A Case Study of Expert Product Design," Des. Stud., **32**(4), pp. 384–415 https://doi.org/10.1016/j.destud.2011.01.003.

[2] Cross, N., 2004, "Expertise in Design: An Overview," Des. Stud., **25**(5), pp. 427–441 https://doi.org/10.1016/j.destud.2004.06.002.

[3] Ahmed, S., Wallace, K. M., and Blessing, L. T., 2003, "Understanding the Differences between How Novice and Experienced Designers Approach Design Tasks," Res. Eng. Des., **14**(1), pp. 1–11 https://doi.org/10.1007/s00163-002-0023-z.

[4] Ullman, D. G., 1992, *The Mechanical Design Process*, McGraw-Hill New York.

[5] Corbett, J., and Crookall, J. R., 1986, "Design for Economic Manufacture," CIRP Ann. Technol., **35**(1), pp. 93–97.

[6] Eberle, B., 1996, *SCAMPER - Games for Imagination Development*, Prufrock Press Inc.

[7] Daugherty, J., and Mentzer, N., 2008, "Analogical Reasoning in the Engineering Design Process and Technology Education Applications.," J. Technol. Educ., **19**(2), pp. 7–21.

[8] Álvarez, A., and Ritchey, T., 2015, "Applications of General Morphological Analysis," Acta Morphol. Gen., **4**(1), pp. 1–40.

[9] Kolodner, J. L., 1992, "An Introduction to Case-Based Reasoning," Artif. Intell. Rev., **6**(1), pp. 3–34 https://doi.org/10.1007/BF00155578.

[10] Lenat, D. B., 1983, "EURISKO: A Program That Learns New Heuristics and Domain Concepts," Artif. Intell., **21**, pp. 61–98.

[11] Daly, S., and Christian, J. L., 2012, "Assessing Design Heuristics for Idea Generation in an Introductory Engineering Course," Int. J. Eng. Educ., **Vol. 28**(2), pp. 1–11.

[12] Kramer, J., Daly, S. R., Yilmaz, S., and Seifert, C. M., 2014, "A Case-Study Analysis of Design Heuristics in an Upper-Level Cross-Disciplinary Design Course," ASEE Annu. Conf. Expo. Conf. Proc.

[13] Campbell, M. I., Cagan, J., and Kotovsky, K., 1999, "A-Design : An Agent-Based Approach to Conceptual Design in a Dynamic Environment," Res. Eng. Des., **11**(3), pp. 172–192.

[14] Cagan, J., Campbell, M. I., Finger, S., and Tomiyama, T., 2005, "A Framework for Computational Design Synthesis: Model and Applications," J. Comput. Inf. Sci. Eng., **5**(3), p. 171 https://doi.org/10.1115/1.2013289.

[15] Moss, J., Cagan, J., and Kotovsky, K., 2004, "Learning from Design Experience in an Agent-Based Design System," Res. Eng. Des., **15**(2), pp. 77–92 https://doi.org/10.1007/s00163-003-0042-4.

[16] Orsborn, S., and Cagan, J., 2009, "Multiagent Shape Grammar Implementation: Automatically Generating Form Concepts According to a Preference Function," J. Mech. Des., **131**(12), p. 121007 https://doi.org/10.1115/1.4000449.

[17] Wood, M., Chen, P., Fu, K., Cagan, J., and Kotovsky, K., 2014, "The Role of Design Team Interaction Structure on Individual and Shared Mental Models," *Design Computing and Cognition '12*, J.S. Gero, ed., Springer Netherlands, Dordrecht, pp. 209–226 https://doi.org/10.1007/978-94-017-9112-0_12.

[18] Sio, U. N., Kotovsky, K., and Cagan, J., 2014, "Analyzing the Effect of Team Structure on Team Performance: An Experimental and Computational Approach," 36th Annu. Conf. Cogn. Sci. Soc., pp. 1437–1442.

[19] McComb, C., Cagan, J., and Kotovsky, K., 2015, "Lifting the Veil: Drawing Insights about Design Teams from a Cognitively-Inspired Computational Model," Des. Stud., **40**, pp. 119–142 https://doi.org/10.1016/j.destud.2015.06.005.

[20] McComb, C., Cagan, J., and Kotovsky, K., 2017, "Optimizing Design Teams Based on Problem Properties: Computational Team Simulations and an Applied Empirical Test," J. Mech. Des., **139**(4), p. 041101 https://doi.org/10.1115/1.4035793.

[21] McComb, C., Cagan, J., and Kotovsky, K., 2016, "Drawing Inspiration From Human Design Teams for Better Search and Optimization: The Heterogeneous Simulated Annealing Teams Algorithm," J. Mech. Des., **138**(4), p. 044501 https://doi.org/10.1115/1.4032810.

[22] Chakrabarti, A., Shea, K., Stone, R., Cagan, J., Campbell, M., Hernandez, N. V., and Wood, K. L., 2011, "Computer-Based Design Synthesis Research: An Overview," J. Comput. Inf. Sci. Eng., **11**(2), p. 021003 https://doi.org/10.1115/1.3593409.

[23] Stöckli, F., and Shea, K., 2017, "Automated Synthesis of Passive Dynamic Brachiating Robots Using a Simulation-Driven Graph Grammar Method," J. Mech. Des., **139**(9), p. 092301 https://doi.org/10.1115/1.4037245.

[24] Schmidt, L. C., and Cagan, J., 1997, "GGREADA: A Graph Grammar-Based Machine Design Algorithm," Res. Eng. Des., **9**(4), pp. 195–213 https://doi.org/10.1007/BF01589682.

[25] Königseder, C., and Shea, K., 2014, "Systematic Rule Analysis of Generative Design Grammars," Artif. Intell. Eng. Des. Anal. Manuf., **28**(3), pp. 227–238 https://doi.org/10.1017/S0890060414000195.

[26] Mitchell, M., 1998, *An Introduction to Genetic Algorithms*, MIT press, Cambridge, MA.

[27] Tai, K., and Akhtar, S., 2005, "Structural Topology Optimization Using a Genetic Algorithm with a Morphological Geometric Representation Scheme," Struct. Multidiscip. Optim., **30**(2), pp. 113–127 https://doi.org/10.1007/s00158-004-0504-y.

[28] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P., 1983, "Optimization by Simulated Annealing," Science (80-.)., **220**(4598), pp. 671–680 https://doi.org/10.1126/science.220.4598.671.

[29] Königseder, C., and Shea, K., 2016, "Visualizing Relations Between Grammar Rules, Objectives, and Search Space Exploration in Grammar-Based Computational Design Synthesis 1," J. Mech. Des., **138**(10), p. 101101 https://doi.org/10.1115/1.4034270.

[30] Königseder, C., Shea, K., and Campbell, M. I., 2013, "Comparing a Graph-Grammar Approach to Genetic Algorithms for Computational Synthesis of PV Arrays," *CIRP Design 2012*, Springer London, London, pp. 105–114 https://doi.org/10.1007/978-1-4471-4507-3_11.

[31] Knight, T., 1998, "Designing a Shape Grammar- Problems in Predictability," Proc. Artif. Intell. Des. '98, pp. 499–516.

[32] Li, X., Schmidt, L. C., He, W., Li, L., and Qian, Y., 2004, "Transformation of an EGT Grammar: New Grammar, New Designs," J. Mech. Des., **126**(4), p. 753 https://doi.org/10.1115/1.1758256.

[33] Triki, E., Collette, Y., and Siarry, P., 2005, "A Theoretical Study on the Behavior of Simulated Annealing Leading to a New Cooling Schedule," Eur. J. Oper. Res., **166**(1), pp. 77–92 https://doi.org/10.1016/j.ejor.2004.03.035.

[34] Falcão, A. F. de O., 2010, "Wave Energy Utilization: A Review of the Technologies," Renew. Sustain. Energy Rev., **14**(3), pp. 899–918 https://doi.org/10.1016/j.rser.2009.11.003.

[35] Kurniawan, A., Pedersen, E., and Moan, T., 2012, "Modelling of Wave Energy Converters Using Bond Graph," Proc. 2012 - 10th Int. Conf. Bond Graph Model. Simulation, ICBGM'12, Part SummerSim 2012 Multiconference, **44**(13).

[36] Kurniawan, A., Pedersen, E., and Moan, T., 2012, "Bond Graph Modelling of a Wave Energy Conversion System with Hydraulic Power Take-Off," Renew. Energy, **38**(1), pp. 234–244 https://doi.org/10.1016/j.renene.2011.07.027.

[37] Drew, B., Plummer, A. R., and Sahinkaya, M. N., 2009, "A Review of Wave Energy Converter Technology," Proc. Inst. Mech. Eng. Part A J. Power Energy, **223**(8), pp. 887–902 https://doi.org/10.1243/09576509JPE782.

[38] N. Ramesh, B., and Arulmozhivarman, P., 2013, "Wind Energy Conversion System - A Technical Review," J. Eng. Sci. Technol., **8**(4), pp. 493–507.

[39] "Ocean Wave Energy," Bur. Ocean Energy Manag. [Online]. Available: https://www.boem.gov/Ocean-Wave-Energy/.

[40] Poullikkas, A., 2014, "Technology Prospects of Wave Power Systems," Electron. J. Energy Environ., (January 2014), pp. 47–69 https://doi.org/10.7770/ejee-V2N1-art662.

[41] McComb, C., Lawson, M., and Yu, Y.-H., 2013, "Combining Multi-Body Dynamics and Potential Flow Simulation Methods to Model a Wave Energy Converter," *1st Marine Energy Technology Symposium* https://doi.org/10.13140/RG.2.1.3817.3285.

[42] McComb, C., 2018, "Towards the Rapid Design of Engineered Systems Through Deep Neural Networks," *Design Computing and Cognition '18*.

[43] McComb, C., Johnson, N. G., Santaeufemia, P. S., Gorman, B. T., Kolste, B., Mobley, A., and Shimada, K., 2018, "Multi-Objective Optimization and Scenario-Based Robustness Analysis of the MoneyMaker Hip Pump," Dev. Eng., **3**(December 2017), pp. 23–33 https://doi.org/10.1016/j.deveng.2018.01.001.

[44] Konigseder, C., and Shea, K., 2016, "Comparing Strategies for Topologic and Parametric Rule Application in Automated Computational Design Synthesis," J. Mech. Des. Trans. ASME, **138**(1), pp. 1–12 https://doi.org/10.1115/1.4031714.