Heuristic-Guided Solution Search Through a Two-Tiered Design Grammar

Lucas Puentes
Department of Mechanical Engineering,
The Pennsylvania State University,
University Park, PA 16802
e-mail: lup93@psu.edu

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

Christopher McComb
School of Engineering Design, Technology, and Professional Programs,
The Pennsylvania State University,
University Park, PA 16802
e-mails: mccomb@psu.edu;
uum209@psu.edu

Grammar-based design is typically a gradual process; incremental design changes are performed until a problem statement has been satisfied. While they offer an effective means for searching a design space, standard grammars risk being computationally costly because of the iteration required, and the larger a given grammar the broader the search required. This paper proposes a two-tiered design grammar that enhances the computational design generation with generalized heuristics to provide a way to more efficiently search a design space. Specifically, this two-tiered grammar captures a combination of heuristic-based strategic actions (often observed in human designers) and smaller-scale modifications (common in traditional grammars). Rules in the higher tier are abstract and applicable across multiple design domains. Through associated guiding heuristics, these macrorules are translated down into a sequence of domain-specific, lower-tier microrules. This grammar is evaluated through an implementation within an agent-based simulated annealing team algorithm in which agents iteratively select actions from either the higher tier or the lower tier. This algorithm is used in two applications: truss generation, which is commonly used for testing engineering design methods, and wave energy converter design generation, which is currently a relevant research area in sustainable energy production. Comparisons are made between designs generated using only lower-tier rules and those generated using only higher-tier rules. Further tests demonstrate the efficacy of applying a combination of both lower-tier and higher-tier rules.
[DOI: 10.1115/1.4044694]

Keyword: computational synthesis

1 Introduction

Exploration of designs within a solution space can be a tedious iterative task for human designers. Alternatively, computational systems can be automated to rapidly navigate through and evaluate problem solutions. The trade-off in automated computational design, however, is that computational systems allow much faster solution search but typically lack the expertise-driven search methodologies, known as heuristics, of human designers. This research aims to reduce this trade-off by providing computational design systems with an approach that incorporates the generalizations of these heuristics. To accomplish this, adaptations are made to the traditional grammar-based solution search, which incrementally alters a design based on a list of predefined valid changes. Our proposed approach divides the list of valid changes, known as a ruleset, into two separate tiers, one that contains traditional incremental design changes and one that contains heuristics-inspired changes.

1.1 Heuristics in Design.

Through experience, human designers often adopt "rules of thumb," which improves their performance and efficacy across the design process. These rules, referred to as heuristics, provide shortcuts or guidance that help the designers efficiently search the design space. In an observational design study, Yilmaz and Seifert [1] demonstrated that heuristic-based strategies allow for a wide variation of solutions to a given problem. They noticed a significant use of design heuristics in experts during concept generation, which enhanced creativity and was beneficial for a novel solution search. Taking advantage of heuristics permits designers to efficiently locate high-quality problem solutions.

Unfortunately, it is unlikely that inexperienced designers have fully developed their own set of personal heuristics for an efficient design generation. Cross [2] noted that experts can process larger quantities of information in a familiar domain than novices. He further remarked that experts possess knowledge which helps them recognize underlying principles in a given design problem, while novices tend to focus on surface features. Similarly, Ahmed et al. [3] saw that novice designers tend to search locally within a problem's design space, as expansive search is limited by their lack of heuristic strategies. Due to a lack of experience-derived strategies, this novice design method resembles a trial-and-error approach. Novices do not fully possess the thoughtfulness and efficient methodology required to make broad searches, therefore constraining design exploration. This contrasts the observed tendencies of experts, who combine sequences of design activities together and execute them sequentially as a heuristic strategy. These heuristic-driven sequences of actions allow for rapid problem-solving across a wide design space area.

Inexperienced human designers can nonetheless take advantage of generalized heuristics for concept generation. Studies have tasked novice designers with incorporating high-level, abstract heuristics into their problem-solving process. For a given design task, Daly and Christian [4] observed a positive correlation between clear heuristic use and a high level of creativity in novice-produced concepts. Concepts that omitted heuristic use were much less-developed and resembled currently existing products and ideas. That study further showed that a heuristic use in beginners created search patterns similar to those seen in experts, as the application of these abstract design changes led designers to considerably distant solution regions of the design space. Kramer et al. [5] expanded on this work showing that there is a potential for generalized heuristics to be applied across various applications. These studies suggest that heuristics can have value even when utilized by those with limited domain experience.

The early stages of design are critical in shaping the final product of the full design process, as later stages build upon prior stages [6,7]. For this reason, studies have analyzed strategies and tools designed

Manuscript received March 24, 2019; final manuscript received August 10, 2019; published online September 5, 2019. Assoc. Editor: Joshua Summers.

Copyright © 2019 by ASME

to assist with initial idea generation. Some studies focus on enhancing broad solution space search [8] and initial design selection [9,10]. Others attempt to narrow the search by making design decisions based on previous solutions to the current problem [11]. Mulet and Vidal [12] provide a set of general heuristic-based guidelines to assist designers with solution concept generation. Comparatively, Chong et al. [13] present an algorithm-powered methodology that can be used to recommend heuristic strategies for concept exploration. Many of these methods are, however, concentrated on searching for new designs through qualitative evaluations as opposed to an algorithmic process. For use in computational design systems, solution search must be represented quantitatively.

Taking inspiration from the benefits that heuristics provide to human designers, this research explores the enhancement of automated design processes through general grammar-based heuristics-driven algorithms. A similar goal was accomplished by the computer program EURISKO [14], which used a heuristic ruleset to compete against human players in the science-fiction role-playing game Traveller. In this competitive game, players design complex spaceship fleets to battle one another. EURISKO gradually learned sets of strategies and heuristics to navigate the spaceship design space, resulting in fleets that greatly outperformed other computational methods, as well as human players, at the time. Later artificial intelligence systems, such as SOAR [15] and ICARUS [16], also incorporated heuristic methods for strategic problem-solving. Sangelkar and McAdams [17] demonstrated a data-mining approach to extract heuristics from design repositories. These extracted heuristics can then serve designers as guidelines for future concept generation in the same design domain.

1.2 Graph Grammars. Graph grammars enable the generation and modification of potential solutions to engineering problems. These grammars are a tool in which physical systems are represented as a layout of connections and nodes containing information on the parameters of a corresponding object [18–21]. Design changes are made through the addition, removal, and manipulation of nodes and connections using defined rules, with the list of all available rules known as a ruleset. Through a sequence of applied rules, solutions evolve to meet the requirements of a specified design problem.

Creation of a graph grammar involves defining the vocabulary as it applies to a design task, establishing a valid ruleset, defining an initial design and then generating designs using the available rules [19]. The language, specifically the vocabulary and ruleset, 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 a rule application, whereas the right-hand side shows the results of the rule application. This left-to-right transformation is implemented within the full design by matching the applied rule to a sub-graph of the current design. Knight [22] 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 that requires solving by said grammar. For the grammars used in the design applications of this paper, rulesets were created with the intention that they could replicate existing designs in the chosen problem domains.

Prior research has applied methods and algorithms to design grammars to improve optimization capabilities [23,24]. One such strategy, simulated annealing, incorporates a dynamic probabilistic method for accepting new design solutions after applications of grammar rules [25]. Early work by Schmidt and Cagan [18] using the graph grammar-based design optimization applied a simulated annealing-driven graph grammar to the cart design using Meccano Set components. The study successfully demonstrated the use of a graph grammar for design optimization and suggested that its implementation allows for complete search within a design space. Recent works utilized graph grammar-based design in gearbox design synthesis [20,26], passive dynamic brachiating robots [21], and optimization of photovoltaic arrays [27], among others. Tai and Akhtar [28] used genetic algorithms, another optimization strategy, to synthesize multiple solutions in a combinatorial manner inspired by evolutionary concepts. Rather than optimizing solutions solely through algorithms, Campbell et al. [29] present a method in which a designer guides the solution interactively through intermittent queries. The work presented in this current paper overcomes the traditional incremental nature of grammars by defining abstract, heuristic-driven meta-rules that can work in concert with traditional rulesets.

1.3 Proposed Grammar Approach. Computational agents are particularly helpful within grammar-based design [30] for rapidly changing and assessing new designs. A limitation of these agents is that they often employ a low-level trial-and-error approach to solution search which resembles that of novice human designers. As a result, human designers must evaluate the opportunity cost associated with establishing and executing these computational systems rather than searching for solutions themselves. However, imbuing these agents with a set of heuristics to guide design changes could greatly increase the efficiency with which they search. To that end, this work proposes a two-tiered approach to the grammar-based design. The lower tier contains domain-specific microrules, which implement singular design changes to an agent's current problem solution. The higher tier contains generalized macrorules, which are translated into a sequence of heuristic-guided and context-sensitive lower-tier rules every time they are applied. These higher-tier rules are purposely abstract, with the intent that they are applicable across multiple design domains. The specific sequence of lower-tier rules derived from an individual higher-tier rule will vary across instantiations and domains, but the guiding heuristics ensure that a similar abstract action will be performed. The purpose of this two-tiered approach is that macrorules apply strategic large-scale solution modifications, while microrules allow for the refinement of solutions. A generalized visualization of this approach can be seen in Fig. 1.

The remainder of this paper is organized as follows. In Sec. 2, the implementation of a two-tiered grammar is described, as well as relevant algorithms used by design agents. Application of the grammar-based approach to truss and wave energy converter (WEC) design problems is explained in Sec. 3. In Sec. 4, the results from the two application areas are presented and analyzed. Comparisons between designs generated using only lower-tier rules and higher-tier rules are examined, along with an additional comparison to designs incorporating a combination of both tiers. Section 5 concludes this paper and details limitations and potential future work regarding two-tiered grammars.

2 Methodology

2.1 Two-Tiered Grammar Rule Approach. At the core of the proposed two-tiered grammar is the notion that higher-tier

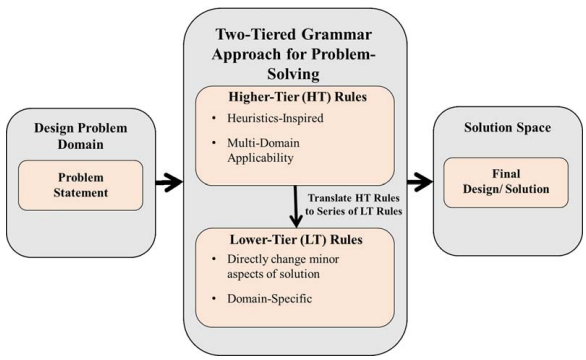

Fig. 1 Generalized implementation of a two-tiered grammar-based approach within a design problem

rules are abstract and heuristics guided, with potential for multi-domain applicability. These higher-tier rules perform as macrorules, executing a sequence of lower-tier rules in a specific routine. Similar to how heuristics enable human designers to make large but informed leaps across a solution space, employing these rules within computational design guides agents to achieve quick, broad search. On the other hand, lower-tier rules are specific to the domain of a given design problem. These serve as microrules, applying small changes to the graph design for local refinement. Together, these two rule tiers aim to explore a broad range of designs in a solution space while also fine-tuning those that satisfy the goals of a problem task.

To apply a higher-tier rule to a design, it must first be translated into a sequence of lower-tier rules, as guided by the heuristic associated with the higher-tier rule. Although higher-tier rules are intended to have utility across multiple domains, there may be instances where some higher-tier rules are fundamentally not applicable in certain domains due to domain limitations. It is up to the grammar designer to determine if a higher-tier rule can be translated into a guided sequence of lower-tier rules or if it is irrelevant for the current domain and should be omitted.

To establish a two-tiered grammar for a given design task, a set of lower-tier rules must first be established. This process follows the same procedure as standard grammar creation: defining the vocabulary as it applies to a design task, establishing a valid ruleset, defining an initial design, and then generating a design [19]. Rules can later be added and modified as required. This grammar creation process specifically defines a class of design problems in which a given state can explicitly change (i.e., transition) to a new state based on a defined set of possible modifications. We refer to this class of problem throughout this paper as *state-transition problems*. Rules developed for these types of problems, such as those established for the two design applications in this paper, should cover all feasible transitions in the given design domain. Tools exist to assist designers with this task, such as the grammar rule analysis method (GRAM) [20].

In formulating a higher-tier ruleset, a list of generalizable heuristics must be recognized. The heuristics chosen should describe macrolevel design changes that are achievable with the existing lower-tier ruleset. In new or unfamiliar problem domains, the optimal heuristic set may be unknown to the designer. However, even a non-optimal heuristic set that leverages designer strategies may still provide benefit in exploring the design space over randomly applied sequences of lower-tier rules. Due to the abstract nature of higher-tier rules, there are likely to be various sequences of lower-tier rules that represent valid implementations of a higher-tier rule in a given domain.

2.2 Assessing Effectiveness of Two-Tiered Grammar. In order to observe the efficacy of individual rule tiers, design repetitions (defined as the full production of a problem solution from start to finish) are conducted using solely one tier at a time. It is hypothesized that using only lower-tier grammar rules may result in slow local improvement of the starting design. Alternatively, using only higher-tier rules may result in infrequent but significant jumps to different designs within the solution space.

To observe the impact of using a combination of both tiers, designs are generated by selecting rules from either tier. Two different rule selection methodologies are tested. The first selects rules randomly from the full list of grammar rules. The second test probabilistically selects a rule based on performance-dependent weights maintained and updated to each rule during solving. Initially, every rule is given equal selection weighting. This initialization behaves identically to the random selection method but differs over time. If the application of a rule improves the solution's quality, the weighting of that rule increases; if the rule worsens the quality, the weighting decreases. This method allows for gradual learning of which rules are effective and ineffective for improving solutions.

Additional simulations are conducted to validate the difference between lower-tier rule sequences implemented by higher-tier rules and sequences formed by random selection. The sequence of lower-tier rules applied when implementing a single higher-tier rule is determined through a guiding heuristic. "Burst algorithms" have been used in grammar-based design applications to apply a sequence of rules in a single-design iteration [26]. Within each rule "burst," multiple rules are applied to the current design, and the solution is evaluated only after the burst is complete. Comparing the performance of higher-tier rules to lower-tier bursts will demonstrate the impact of heuristically guiding a sequence of lower-tier rules rather than randomly selecting them. To test this lower-tier burst method, solutions are generated by applying three random lower-tier rules per each design iteration. Because the lower-tier sequence length for higher-tier rules can be dependent on the current design (in such cases where a higher-tier rule makes changes to the entire system) and limits determined by the designer (such as restricting how much the system can expand in one higher-tier rule application), determining a definite average length is not feasible. A burst length of three is chosen as an estimate of this sequence length.

2.3 Agent-Based Search Methodology for Two-Tiered Grammar

2.3.1 Heterogeneous Simulated Annealing Teams. The current study uses the heterogeneous simulated annealing teams (HSAT) framework as an optimization algorithm for testing our two-tiered grammar [31]. This framework has shown success in applications similar to those planned in this study. The HSAT framework employs a team of computational agents that engage in a locally sensitive search and quality-informed solution sharing. These two characteristics allow for both broad and deep solution search. Locally sensitive search is achieved through an agent's use of an adaptive simulated annealing algorithm for design change acceptance. After a design change has been applied to an agent's current solution, the new candidate solution is evaluated, and its results are compared with the current solution's results. If the design is evaluated as an improvement, then the agent adopts this candidate 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):

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

where p is the probability that candidate solution and x_{new} will be accepted as the replacement to the 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 schedule [32]. The Triki schedule calculates the temperature of an agent given index i through Eq. (2):

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

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 perform quality-informed solution sharing, the second main characteristic of HSAT. The current solution for each agent is acquired and stored within the vector **F,** Eq. (3):

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

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 $\mathbf{F}$ is compared with the worst current solution in order to create the weight vector $\mathbf{W}$, Eq. (4):

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

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):

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

where the "mult" function designates that the agent is selected from an applied multinomial distribution formed by a proportional weighting of each solution within the weight vector $\mathbf{W}$.

The designer selects the collaboration frequency of agent teams 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. There is a trade-off between high and low frequencies, in that the higher frequencies are likely to result in a more locally refined design, while lower frequencies are likely to span globally across the solution space with less refinement. Because this solution sharing is performed probabilistically rather than deterministically, agents may not necessarily select the current top solution and can instead continue exploring alternative designs. For the design problems 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 toward a final design. After the collaboration, each agent continues to iterate through design solutions until either the next collaboration period or the endpoint of the design process. At this point, a final design is selected by the team.

2.3.2 HSAT Applied to Two-Tiered Grammars. A computational agent team is organized according to the HSAT framework to apply the two-tiered grammars to the design applications in this study. Each application has a unique objective function to evaluate a solution, and the agents are given the goal of minimizing this value. The agent team performs a number of total iterations (individual rule applications and objective function evaluations) for every design repetition (the complete generation of a solution from start to finish), as per McComb et al. [31]. Each agent is instantiated with the same rule-tier selection method and solution evaluation method. At the beginning of each agent's individual iteration, the agent probabilistically chooses which rule tier to select using its rule-tier weighting values. 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 applications in this work, designs generated by using a single rule tier are first compared. Agent teams are configured to only select rules from a single tier for the entirety of their design process. Once a tier is chosen, the agents stochastically apply a rule from that tier. Additional tests examine the performance of combined-tier selection methods, in which all rules from both tiers are combined. The implementation of these methods is explained further in Sec. 3. The selected rule is applied to the design and the agent evaluates this new candidate design. If the new design is valid (i.e., physically realizable), the agent decides whether to reject or accept the change in accordance with the simulated annealing methodology. Across both applications, each agent independently updates their temperature every 10 iterations using the Triki scheduling in Eq. (2). The initial temperature and Triki parameter chosen for each agent were based on values used previously in the HSAT algorithm [31], but scaled in magnitude to better fit the specific application. Because this initial temperature and cooling rate are not specifically optimized to match the agent's rule-tier preferences, the same values are used in all preference settings.

In accordance with the HSAT framework, agents also intermittently interact via quality-informed solution sharing. For this study, agents collaborate after every 10 iterations, allowing agents to attempt multiple rule applications before considering adopting another's design. In instances where higher-tier rules are used, collaboration between computational agents is highly important because most higher-tier rules will lead to design divergence. By nature of the higher-tier rules, a single rule application can greatly alter the design into different and distant regions of a solution space. This divergent behavior aligns with observations made by Daly and Christian [4], 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 passed and the objective function values of all team agents has converged, a final collaboration period selects a final design solution.

For a single design, three computational agents are instantiated and arranged according to the HSAT. These three agents model a three-member design team in which all members work toward solving the same state-transition design problem. The use of three agents for this work is selected to enable the exploration of multiple potentially divergent solutions, while not significantly increasing the computational time needed to produce a single final design. The same initial design is used across all repetitions and rule tier preferences tested. After each agent has completed one repetition of 500 iterations (chosen in this implementation to allot simulations adequate time to demonstrate rule tier effectiveness), the agent team collaborates once more and conveys its final design. This full design process is repeated for 100 repetitions in this work.

3 Applications

The proposed two-tiered grammar approach is applied to two separate state-transition design problems, with both requiring configuration optimization. State-transition problems can highlight the behavioral differences between the two grammar rule tiers regardless of the complexity of the represented design domain. Across all complexities, lower-tier rules consistently perform single-step, minor design changes, and higher-tier rules implement heuristically guided sequences of those lower-tier rules. The first problem is the configuration design of a truss for a specified load-bearing context. Problems of this type are commonly used for testing the effectiveness of traditional grammar-based design algorithms. Second is the design of a WEC for operation in an ocean environment. This task is not only a challenging engineering design problem but also one of current societal relevance. Although the lower-tier rules vary across the domains of both applications, the same set of higher-tier rules is used to demonstrate their generalizability. The five higher-tier rules established for these applications are as follows: *H1: Increase design complexity*, *H2: Decrease design complexity*, *H3: Change design scale*, *H4: Replicate pattern*, and *H5: Standardize*. These higher-tier grammar rules are adapted from a small number of design heuristics identified in an extensive list by Daly and Christian [4].

3.1 Truss Design

3.1.1 Truss Design Background. The proposed grammar-based approach is first applied to truss generation for a load-bearing problem. Truss design problems have been frequently used in prior research on engineering design methods [23,24,26,33–35]. Structural design problems of this type are commonly used for verifying new algorithms because they are well understood and can be quickly and clearly evaluated. For this application, a truss is generated which must support an applied load with specified support locations. Only axial stress and buckling failures are considered in design assessment.

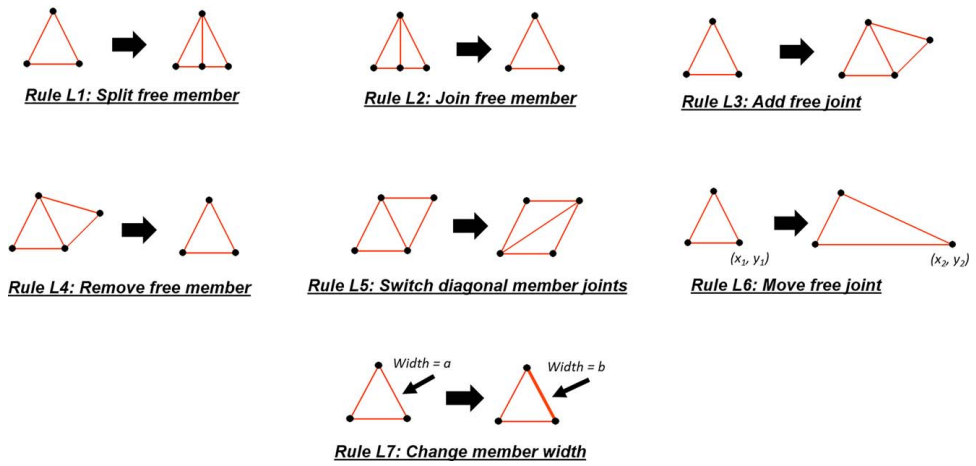

Fig. 2 Lower-tier graph grammar rules for truss design domain

3.1.2 Truss Grammar Creation. The lower-tier rules of the truss grammar are inspired by rules used in prior work [26]. These rules are capable of applying topologic, spatial, and parametric changes to a design solution. All lower-tier truss grammar rules can be found in Fig. 2. The higher-tier ruleset is chosen as a sample of heuristics that enable large-scale modifications typical in configuration design. All higher-tier truss grammar rules are shown in Fig. 3. For future design problems, additional rules may be added to either rule tier if required.

3.1.3 Evaluating Truss Solutions. The goal of the truss design task is to produce a design that can support an applied load with a set of designated fixed nodes. Computational agents accomplish this through to the minimization of the objective function:

$$Q = m + 100 \cdot \max(0, f_t - \min(f_s))^2 \tag{6}$$

where Q is the objective function value, m is the total mass, f_t is the target factor of safety for all members in the design, and $\min(f_s)$ is the lowest factor of safety for all members in the design. In essence, this objective function is the mass of the truss plus a penalty if the truss has a poor factor of safety. After each rule is applied to the solution, the agent determines acceptance of the design change using its simulated annealing algorithm. Each individual agent operates and updates its own simulated annealing algorithm.

3.2 Wave Energy Converter Design

3.2.1 Wave Energy Background. Renewable and sustainable energy methods have long been of great interest to the engineering research community. Energy from ocean waves has been estimated to possess the highest energy density and potential active power generation time among renewable resources [36]. WECs are devices that can be used to harvest energy from ocean waves. Because WECs are still an emerging technology, there has been no convergence on a standard WEC design [36], in part because WEC effectiveness is highly dependent on location-based factors (e.g., ocean depth, typical wave frequency). For this reason, engineers must explore multiple potential designs for a single situation.

An example of two dissimilar converter topologies can be seen in the Pelamis device and Ocean Power Technology's PowerBuoy device [36,37]. The Pelamis, tested off the shores of Scotland, is a snake-like chain of four cylinders which generates power through the rotational motion of the cylinders about its joints. Alternatively, the PowerBuoy, also tested off the coast of Scotland, converts wave energy through the linear motion between a large circular floating disk and a submerged buoy. Figure 4 demonstrates

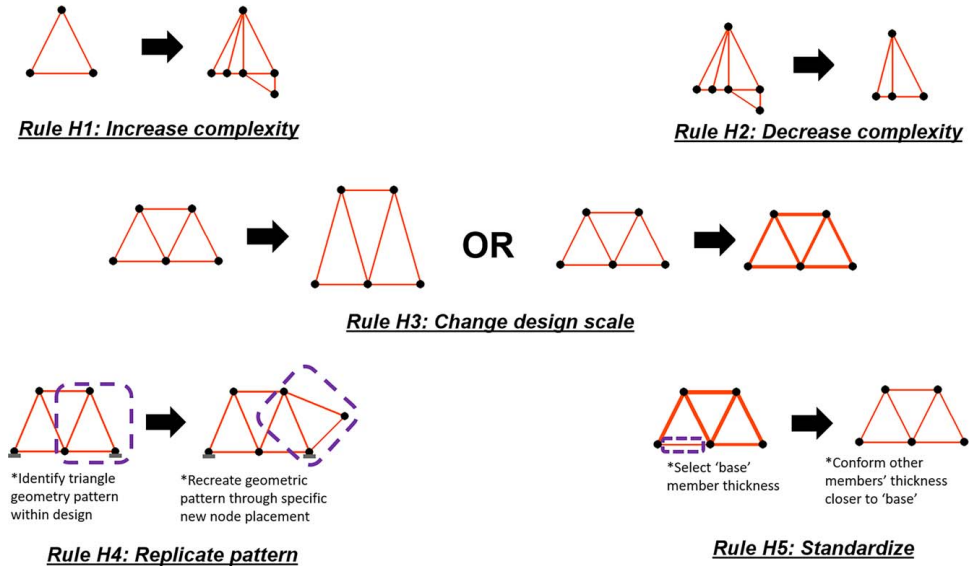

Fig. 3 Higher-tier graph grammar rules for truss design domain

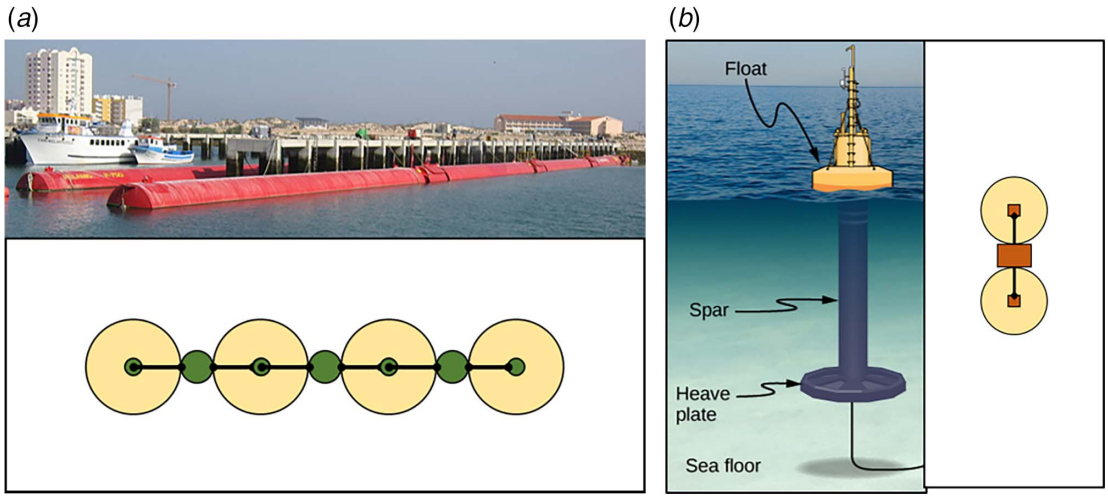

Fig. 4 Representation of (*a*) the Pelamis device (Permission to use from Dipl. Ing. Guido Grassow, granted under Creative Commons (CC BY-SA 3.0)) and (*b*) Ocean Power Technology's Powerbuoy (Permission granted under Creative Commons (CC BY 4.0)) within the graph grammar developed for this study

how these example WECs can be portrayed through the graph grammar representation introduced in Sec. 3.2.2. Exploring these and other WEC designs through a computational approach has the potential to rapidly advance the performance of WECs. For this reason, a WEC design problem is selected as a case study for implementation of this paper's two-tiered grammar rule approach. It is assumed that the system is not anchored to the ocean floor and that all floating bodies are spherical.

3.2.2 Wave Energy Converter Grammar Creation. WECs contain one or more power take-off units (PTOs) that transform kinetic energy from ocean waves into electrical power [37,38]. PTOs act as a joint that connects two floating bodies and generate power through the relative motion of those bodies. Most PTOs fall into two general classifications, depending on the mode that is used to extract power: rotational PTOs that generate electrical energy from the rotational motion of the bodies about the joint and linear PTOs that generate electrical energy from the relative translational motion between bodies [39].

Unlike the truss design application, a grammar for the wave energy converter design was not readily available in the literature. A lower-tier grammar ruleset is first established for the WEC domain, derived from examination and exploration of existing WEC device designs. These rules allow for addition, removal, relocation, and parameter adjustment of floating bodies, linear PTOs, and rotational PTOs. This grammar is capable of recreating a broad array of existing devices (for two examples, see Fig. 4). The full lower-tier ruleset is shown in Fig. 5. Next, the same higher-tier ruleset used within the truss domain is adapted for the WEC domain. This demonstrates the broad applicability of higher-tier rules across multiple domains. This adapted set of higher-tier rules, as applicable to the WEC grammar, can be found in Fig. 6.

3.2.3 Evaluation of Approach Based on the Wave Energy Converter Design Problem. The performance of WEC designs is assessed using a simulation that combines time- and frequency-domain methodologies [40,41]. These simulations calculate both the power output and mass of the device. The objective function of this design problem (Eq. (7)) aims to minimize the negative of the device's power density:

$$Q = -P/m \tag{7}$$

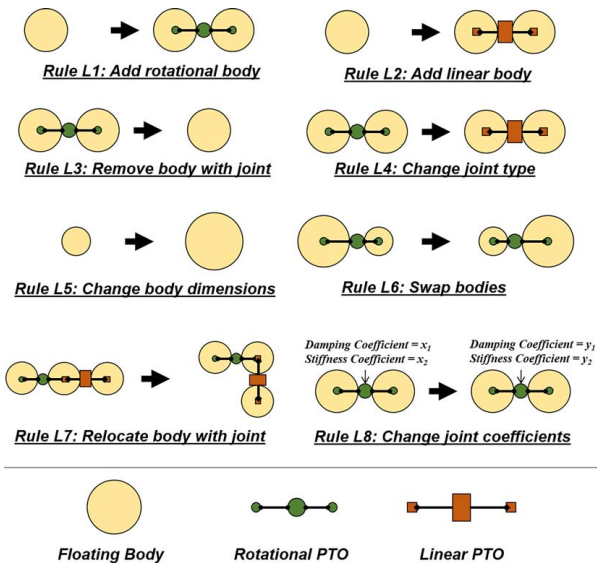

Fig. 5 Lower-tier graph grammar rules for wave energy converter design domain

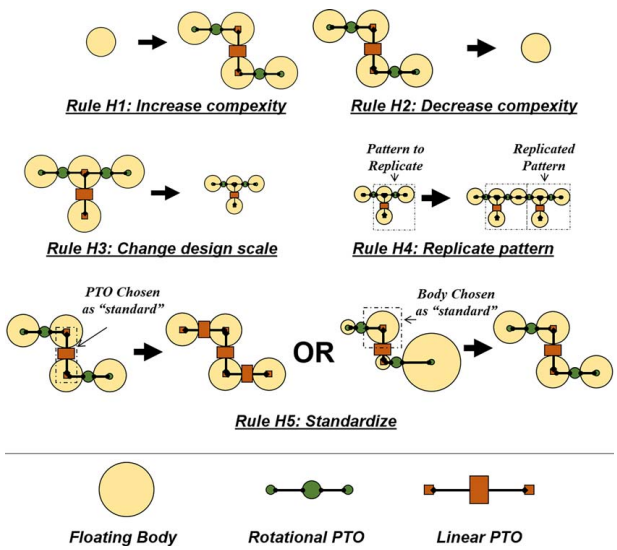

Fig. 6 Higher-tier graph grammar rules for wave energy converter design domain

That is, as the design's power output P divided by its mass m increases, the objective function value Q of the design is improving. Here, mass is a proxy for cost, which has been shown to yield similar optima as direct cost minimization [42]. The goal of the agent team, therefore, corresponds to producing a device that generates the most power for the least possible cost.

The mass of a floating body is found by multiplying its density by its weight. The total system mass is simply a sum of each individual floating body mass. The total power of the system is dependent on the power produced in all PTOs. The power of an individual PTO is based on the motion of the bodies to which it is connected, while PTO type determines which of two possible power equations to use. Rotational PTOs use Eq. (8) to calculate instantaneous power P (in W)

$$P = \kappa(|\omega_A - \omega_B|)^2 \tag{8}$$

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 use Eq. (9) to calculate power:

$$P = k(|v_A - v_B|)^2 \tag{9}$$

where k is the linear damping coefficient of the PTO (in N s/m) and v is the linear velocity of bodies A and B (in m/s). These equations provide a measure of the instantaneous power output of the WEC. Averaging these values across PTOs and across timesteps of the simulation yields average power production for the full system.

4 Results and Discussion

4.1 Truss Configurations. Figure 7 shows the objective function value of several variations of the agent-based algorithm on a logarithmic scale. The variations include one using only lower-tier rules, one using only higher-tier rules, and two variants that employ randomized selection between the two rule tiers. Designs generated with only higher-tier rules make swift solution improvements but soon converge to an average objective value of approximately 150. In the context of the design problem, the higher-tier rules are able to meet the factor of safety requirement of the truss in only a few rule applications. Later in the design, this ruleset struggles to reduce mass without over-sacrificing strength. As these rules apply multiple changes in a single iteration and are somewhat stochastic, refinements that are aligned with desired changes in the objective functions are unlikely.

In direct contrast, designs generated with only lower-tier rules are significantly slower in minimizing the objective function but are ultimately more effective for fine-tuning the system. This ruleset steadily and carefully improves the design, preventing the solution from converging prematurely. While the lower tier is slow to achieve the desired factor of safety, it can easily reduce mass afterward without over-reducing strength. Therefore, if a valid solution is required very quickly, higher-tier rules are a better choice than lower-tier rules. However, if a solution search time is not a concern, then the lower-tier rules should be used. The early inefficiency of the lower-tier and the later limitations of the higher-tier suggest that a combination of the two tiers may compensate for the shortcomings of both.

For combined-tier design generation approaches, the rate of objective function minimization in early iterations is between designs using solely lower-tier or higher-tier rules. The access to the higher-tier ruleset enables these designs to make the same initial improvement leaps that designs produced with higher-tier rules demonstrated. These leaps provide designs with an advantageous start over designs produced with lower-tier rules. Additionally, these designs do not prematurely converge like designs produced using only higher-tier rules. They instead continue to gradually improve the solution well into the later design iterations. A combined-tier approach produces a lower average objective function value than designs produced with lower-tier rules, suggesting that higher-tier rules can still be effective later in the design if used in conjunction with the lower tier.

Due to similar initial conditions between the random and the probabilistic selection methods, both methods perform the same in early iterations. Over time, probabilistic selection shifts the weighting assigned to each rule to prefer selecting those which were helpful previously. For this design problem, probabilistic selection demonstrates a trend of rapid solution improvement slightly longer than random selection. After this period, however, both methods converge to a similar value. This difference in trends can be attributed to a shift in the effectiveness of specific grammar rules throughout the generation process. The probabilistic method will continue to select rules that improved the solution early in the design process until these rules become less effective. The likelihood of selecting these specific rules will decrease until the method again resembles that of random selection.

By design, higher-tier rules employ a guided sequence of lower-tier rules. These sequences differ greatly from those which randomly select rules, which can be modeled as lower-tier bursts. Solutions generated by applying bursts of three randomly selected lower-tier rules initially follow the same objective function trend as designs produced with lower-tier rules. But as seen in Fig. 8, these bursts lose effectiveness rather quickly and the solution improvement rate greatly decreases. As noted previously, a burst length of three rules was chosen as the expected average of lower-tier rules implemented by a single higher-tier rule. These lower-tier

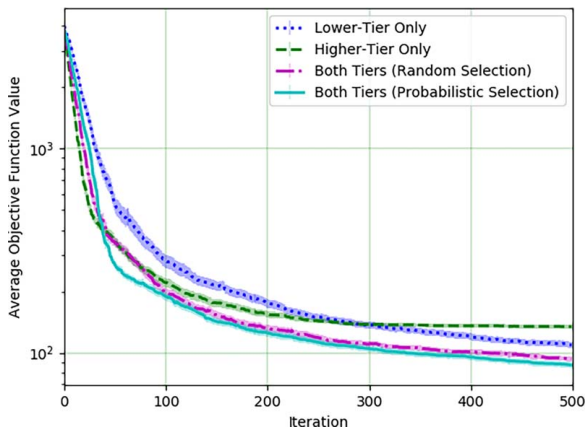

Fig. 7 Comparison of average objective function value for truss design problems over agent team iterations using various rule tier preferences (shaded regions show ±1 standard error)

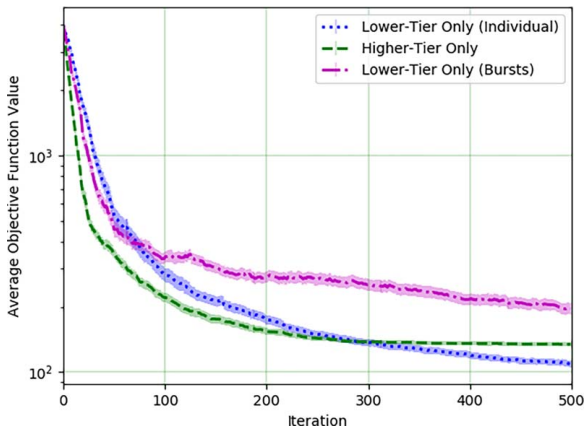

Fig. 8 Comparison of average objective function value for truss design problem over agent team iterations using burst algorithms and singularly applied rules (shaded regions show ±1 standard error)

burst designs gradually minimize the objective function, but ultimately do not reach even the same convergence value as designs produced with higher-tier rules at the end of 500 iterations.

Since each burst applies an expected average length of the lower-tier rule sequence stemming from a higher-tier rule, the value of the higher-tier rules must not be merely in the application of lower-tier rules. Rather, the correct rules are applied in the correct order. This enables an effective heuristic-guided search, and the importance of proper sequencing has been demonstrated in other work involving truss design [43,44]. Across each rule application, the heuristics associated with each higher-tier rule remain constant, and thus so does the general sequences of implemented lower-tier rules. Random lower-tier bursts produce rule sequences that are very diverse and consequently unlikely to resemble any higher-tier rule. Further, by performing multiple unguided changes in a single-design iteration, there is a chance that some changes can counteract the improvements of another change. In comparison, higher-tier rules are less likely to behave in this manner, as the multiple changes are intended to complement one another through the guiding heuristic.

Figure 9 displays which individual rules were accepted in different stages of the design generation. Stages are divided into groups of 100-iteration chunks, with each chunk showing the proportion of all accepted rules in that stage. The likelihood of accepting specific individual rules from either tier shifts as agents evolve a design. Certain rules become more beneficial for improving the solution in later design iterations, while others become ineffective. Rules *HT3: Change design scale* and *HT5: Standardize* are the only higher-tier rules that continue to make accepted changes after the first hundred iterations. By this point, other higher-tier rules are mostly unable to further improve the solution, and therefore, the changes they make are unlikely to be accepted. Similarly, the lower-tier rules *LT4: Remove free member*, *LT6: Move free joint*, and *LT7: Change member width* stay prominent across all iterations. All other lower-tier rules gradually become ineffective for solution refinement. The presence of accepted rules from both rule tiers in later iterations further supports the impression that the tiers complement one another in preventing premature solution convergence.

Generally, for grammars defining state-transition design problems, most rules can be categorized into three universal groups: topological, spatial, and parametric. Topological rules alter the total number of components within a design. Spatial rules reorganize and reposition components. Parametric rules adjust design values within the components themselves. Every rule from both tiers can be classified into one of these groups. Rules from all three categories show signs of use in early design stages. Over the course of the design process, rules within the topological group tend to become less prominent, with the exception of *LT4: Remove free member*, which actually increases in use. The heavy objective function penalty given to solutions which do not meet the factor of safety requirement likely leads to complex, over-strengthened early designs. From this point, agents search for which members to remove without sacrificing strength. The full set of parametric rules for the truss grammar rules *LT7: Change member width, HT3: Change design scale,* and *HT5: Standardize*, are all among the most accepted rules in the later design iterations. This suggests that rules focused on refinement, whether lower or higher tier, are generally more effective for late-design improvement than those which alter topology or spatial organization. The exception to this is the previously mentioned *LT4: Remove free member*. With the ability to change rule selection methods throughout the design process, there is potential for agents to identify effective times to use each rule group.

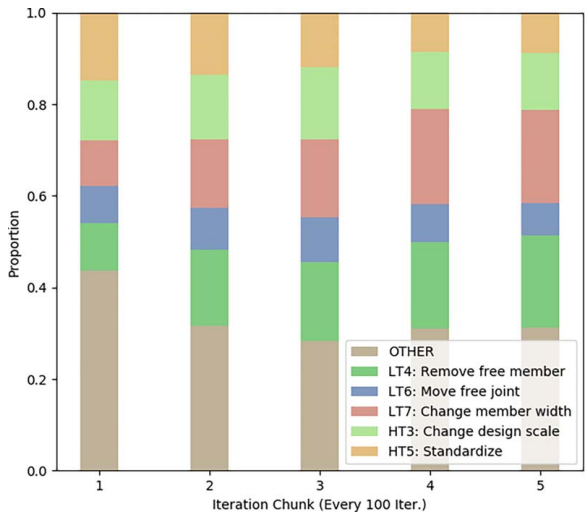

Fig. 9 Proportionality of individual rules accepted into design across truss design process (by random selection from both rule tiers). Iterations are lumped into 100-iteration chunks.

4.2 Wave Energy Converter Configuration. Solution improvement trends examined in the WEC design problem generally match those seen in the truss design problem. Figure 10 shows that higher-tier solutions demonstrate a prompt minimization in objective function value but converge to approximately −450 W/kg by the 100th iteration. After convergence, higher-tier rules cannot perform the small-scale changes that the design requires for further improvement. Alternatively, lower-tier solutions do not reach the same quality until almost the 200th iteration. These designs continue to improve to almost −700 W/kg at the end of 500 iterations. For quick valid solution generation, higher-tier rules are effective, but for longer-term design improvement, lower-tier rules allow for more effective refinement.

For designs produced with both rule tiers, solution improvement initially follows the same trend as designs produced with higher-tier rules. However, both combined-tier selection methods significantly surpass the higher-tier convergence point, reaching between −900 and −1000 W/kg by the end of 500 design iterations. The exceptional performance of these designs is likely due to the manner in which the two tiers complement one another. Early in the solution, higher-tier rules can make consistently large leaps toward improved designs. While these rules would plateau on their own, the lower-tier rules make small-scale adjustments that allow the higher tier to continue assisting the design. As the designs produced with lower-tier rules do not appear to have yet converged to a final design, many more design iterations would likely be necessary to reach the same objective function value as designs produced with both rule tiers.

Lower-tier burst designs show behavior similar to the use of singularly applied lower-tier rules, as seen in Fig. 11. This is unlike the

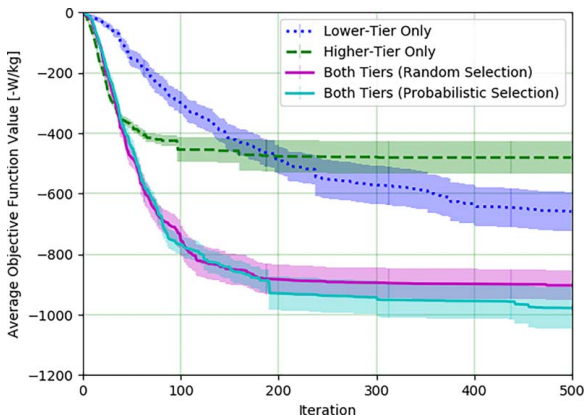

Fig. 10 Comparison of average objective function value for WEC design problems over agent team iterations using various rule tier preferences (shaded regions show ±1 standard error)

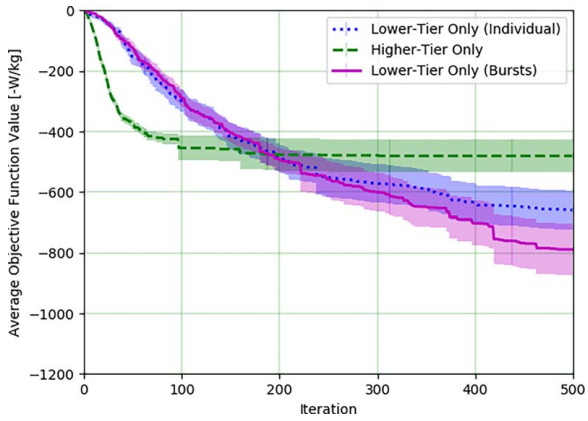

Fig. 11 Comparison of average objective function value for WEC design problem over agent team iterations using burst algorithms and singularly applied rules (shaded regions show ±1 standard error)

trend observed in the truss design problem, in which the burst designs were far less refined than singularly applied designs. For this specific design problem, quality improvement may be less sensitive to design change sequences. Whereas the objective function of the truss problem applied large penalties for not satisfying a minimum strength requirement, WEC design quality is not severely impacted if a single evaluation metric becomes inferior. The lower-tier burst WEC designs ultimately surpass the convergence point of designs produced with higher-tier rules. Again, lower-tier bursts are unpredictable and cannot reliably be used to replace the purpose of heuristically driven higher-tier rules

Individual rule acceptance across the full WEC design process, shown in Fig. 12, shows a common pattern with that of truss design: individual rule acceptance shifts as the design is improved and is linked to the rule's categorization. Although unlike the truss design, only rules focused on parametric changes were effective for late-design refinement. The parametric-based rules *LT8: Change joint coefficients*, *HT3: Change design scale*, and *HT5: Standardize* account for almost 60% of accepted rules in the first hundred iterations but increase to 80% in subsequent iterations. The spatial rule *LT7: Relocate body with joint* accounts for nearly all of the remaining 20% in the later iterations. These rule category trends strengthen the idea that allowing agents to select rules from specific groups depending on the progression point of the problem solution may improve their search efficiency. The optimal rule groups for a given progression point can vary across design problems, as observed with the difference in late-design refinement

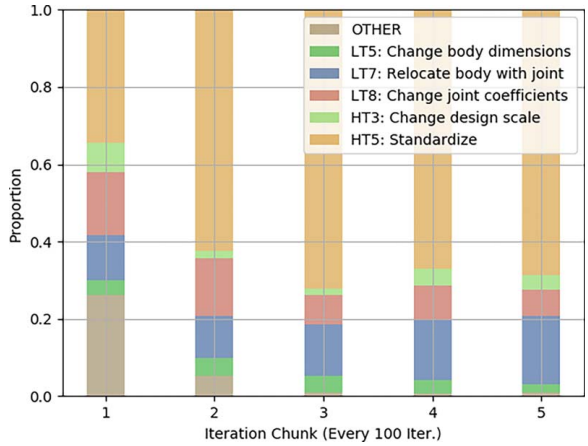

Fig. 12 Proportionality of individual rules accepted into design across WEC design process (by random selection from both rule tiers). Iterations are lumped into 100-iteration chunks.

between truss and WEC designs. Therefore, any prescriptive rule group selection approaches pursued in the future must be sensitive to the characteristics of the problem being solved.

A limitation of this study is in the development of the grammar rules themselves. The inclusion of any new rule in a grammar ruleset has an unpredictable impact, even when that ruleset is used by experienced designers rather than computational algorithms. Grammar creators need to design rules carefully, 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. Tools exist to aid the creation of fully developed grammars such as the GRAM [20]. GRAM provides feedback to grammar developers as a guide for considering new, omitting unnecessary, and altering inefficient rules. Additional methods, such as that presented by Orsborn et al. [30], attempt to automatically derive grammar rules by statistically analyzing valid designs, reducing the need to rely on a designer's personal experience in the given domain. Using similar tools may help ensure more refined development of both the lower- and higher-tier rulesets within a design domain.

5 Conclusions

This work presents a heuristic-inspired, two-tiered grammar approach to computational design. Teams of computational agents iteratively apply graph grammar rules from one of two rule tiers. The domain-specific lower tier is composed of microrules that apply single-step transformations to an agent's current design. The higher tier consists of generalizable macrorules that apply a sequence of heuristically guided lower-tier rules. This proposed methodology is applied to state-transition design problems for trusses and wave energy converters, with configuration optimization as the goal for each. Although these specific problems are designed at a conceptual level, the effectiveness of the two-tiered grammar approach is expected to be similar across varied and more complex state-transition design problems. Regardless of problem complexity, lower-tier rules consistently perform minor, single-step design changes and higher-tier rules implement heuristically guided sequences of those lower-tier rules. Problem completion is achieved through the HSAT optimization framework. As the baseline, HSAT implementation functions with only a typical, lower-tier grammar ruleset, the HSAT implementation that utilizes the two-tiered grammar facilitates a direct comparison between the performance of single-tiered and two-tiered grammar-based approaches. Alternative optimization methods could also be used to make this same comparison of grammar-based approaches, and this should be a subject of future work.

By constructing solutions using only one rule tier, it is observed that designs produced with higher-tier rules can make initial design improvements more effectively than designs produced with lower-tier rules. Over a longer timeframe, however, higher-tier rules reach a premature convergence point, and lower-tier rules can ultimately produce a better solution. These higher-tier rules are guided by abstract heuristic strategies and thus require less solution evaluations to search the initial design space more broadly. Applying bursts of unguided lower-tier rules in each design iteration exhibits similar behavior to singular lower-tier rules early in the process, indicating that unguided lower-tier sequences are unlikely to improve an early solution as rapidly as a guided sequence. In both design applications, the combined use of both rule tiers leverages the initial quick improvements possible with higher-tier rules as well as later optimization afforded by lower-tier rules. At the very least, higher-tier rules could be beneficial for boosting the early performance of traditional algorithms that primarily employ lower-tier rules. This set of heuristically inspired actions begins to empower computational design agents with the benefits that human designers derive from heuristics. These general macrorules are fundamentally designed to guide a sequence of actions in a

similar style across multiple domains. As the capabilities of computational design systems are enhanced through methods such as those explored in this work, the benefits of setting up and implementing such a system may begin to outweigh the costs of doing so.

Future work may proceed along a variety of potential avenues. For instance, selection algorithms can be enhanced through deeper analysis on when to apply specific rules and rule categories. By imbuing agents with more intelligent algorithms, they may be able to assess opportune times to make specific design changes. An additional route for expanding upon this work would be further development in choosing which generalizable heuristics are applicable for a given design problem. An oversaturation of unnecessary rules has the potential to reduce the chance of selecting rules that are beneficial to design improvement, requiring more exploration of the available ruleset before exploitation can take place. Permitting the agent to select its own higher-tier ruleset as part of its augmented iterative rule selection process may lower the risk of oversaturation while allowing the rules to maintain broad applicability. Lastly, the results of this work can be further confirmed through the application of the presented grammar approach with additional configuration design problems or extended to other types of state-transition problems. In doing so, differences in domain characteristics (e.g., problem type, grammar rule set size) can be explored.

Acknowledgment

This material is based upon work supported by the Defense Advanced Research Projects Agency (Funder ID: 10.13039/100000185) through cooperative agreement N66001-17-1-4064 and the Air Force Office of Scientific Research (Funder ID: 10.13039/100000181) 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. An earlier version of this paper was presented at the 2018 ASME IDETC Design Automation Conference [45].

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.
[2] Cross, N., 2004, "Expertise in Design: An Overview," Des. Stud., **25**(5), pp. 427–441.
[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.
[4] Daly, S., and Christian, J. L., 2012, "Assessing Design Heuristics for Idea Generation in an Introductory Engineering Course," Int. J. Eng. Educ., **28**(2), pp. 1–11.
[5] Kramer, J., Daly, S. R., Yilmaz, S., and Seifert, C. M., 2015, "A Case-Study Analysis of Design Heuristics in an Upper-Level Cross-Disciplinary Design Course," ASEE Annual Conference and Exposition, Indianapolis, IN, June 15–18, pp. 24.23.1–24.23.17.
[6] Corbett, J., and Crookall, J. R., 1986, "Design for Economic Manufacture," CIRP Ann. Technol., **35**(1), pp. 93–97.
[7] Ullman, D. G., 1992, *The Mechanical Design Process*, McGraw-Hill, New York.
[8] Eberle, B., 1996, *SCAMPER—Games for Imagination Development*, Prufrock Press Inc., Waco, TX.
[9] 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.
[10] Álvarez, A., and Ritchey, T., 2015, "Applications of General Morphological Analysis," Acta Morphol. Gen., **4**(1), pp. 1–40.
[11] Kolodner, J. L., 1992, "An Introduction to Case-Based Reasoning," Artif. Intell. Rev., **6**(1), pp. 3–34.
[12] Mulet, E., and Vidal, R., 2008, "Heuristic Guidelines to Support Conceptual Design," Res. Eng. Des., **19**(2–3), pp. 101–112.
[13] Chong, Y. T., Chen, C.-H., and Leong, K. F., 2009, "A Heuristic-Based Approach to Conceptual Design," Res. Eng. Des., **20**(2), pp. 97–116.
[14] Lenat, D. B., 1983, "EURISKO: A Program That Learns New Heuristics and Domain Concepts," Artif. Intell., **21**(1–2), pp. 61–98.
[15] Laird, J., Newell, A., and Rosenbloom, P. S., 1987, "SOAR : An Architecture for General Intelligence," Artif. Intell., **33**(1), pp. 1–64.
[16] Langley, P., McKusick, K. B., Allen, J. A., Iba, W. F., and Thompson, K., 1991, "A Design for the ICARUS Architecture," ACM SIGART Bull., **2**(4), pp. 104–109.
[17] Sangelkar, S., and McAdams, D. A., 2013, "Mining Functional Model Graphs to Find Product Design Heuristics With Inclusive Design Illustration," ASME J. Comput. Inf. Sci. Eng., **13**(Dec.), pp. 1–11.
[18] Schmidt, L. C., and Cagan, J., 1997, "GGREADA: A Graph Grammar-Based Machine Design Algorithm," Res. Eng. Des., **9**(4), pp. 195–213.
[19] 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," ASME J. Comput. Inf. Sci. Eng., **11**(2), p. 021003.
[20] Königseder, C., and Shea, K., 2014, "Systematic Rule Analysis of Generative Design Grammars," Artif. Intell. Eng. Des. Anal. Manuf., **28**(3), pp. 227–238.
[21] Stöckli, F., and Shea, K., 2017, "Automated Synthesis of Passive Dynamic Brachiating Robots Using a Simulation-Driven Graph Grammar Method," ASME J. Mech. Des., **139**(9), p. 092301.
[22] Knight, T., 1998, "Designing a Shape Grammar: Problems of Predictability," *Artificial Intelligence in Design '98*, Springer, Dordrecht, pp. 499–516.
[23] Reddy, G., and Cagan, J., 1995, "An Improved Shape Annealing Algorithm For Truss Topology Generation," ASME J. Mech. Des., **117**(2A), pp. 315–321.
[24] Shea, K., and Cagan, J., 1997, "Innovative Dome Design: Applying Geodesic Patterns With Shape Annealing," Artif. Intell. Eng. Des. Anal. Manuf., **11**(5), pp. 379–394.
[25] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P., 1983, "Optimization by Simulated Annealing," Science, **220**(4598), pp. 671–680.
[26] Königseder, C., and Shea, K., 2016, "Visualizing Relations Between Grammar Rules, Objectives, and Search Space Exploration in Grammar-Based Computational Design Synthesis," ASME J. Mech. Des., **138**(10), pp. 1–11.
[27] 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 Conference, Bangalore, India, Mar. 28–31, pp. 105–114.
[28] 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.
[29] Campbell, M. I., Rai, R., and Kurtoglu, T., 2012, "A Stochastic Tree-Search Algorithm for Generative Grammars," ASME J. Comput. Inf. Sci. Eng., **12**(Sep.), pp. 1–11.
[30] Orsborn, S., and Cagan, J., 2009, "Multiagent Shape Grammar Implementation: Automatically Generating Form Concepts According to a Preference Function," ASME J. Mech. Des., **131**(12), p. 121007.
[31] 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," ASME J. Mech. Des., **138**(4), p. 044501.
[32] 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.
[33] Berge, N., and Campbell, M. I., 2016, "Optimal Linkage Shapes of Planar Mechanisms Using Topology Optimization," Volume 4A: Dynamics, Vibration, and Control, Phoenix, AZ, Nov. 11–17, pp. 1–7.
[34] Hooshmand, A., and Campbell, M. I., 2016, "Truss Layout Design and Optimization Using a Generative Synthesis Approach," Comput. Struct., **163**(Jan.), pp. 1–28.
[35] McComb, C., Cagan, J., and Kotovsky, K., 2017, "Optimizing Design Teams Based on Problem Properties: Computational Team Simulations and an Applied Empirical Test," ASME J. Mech. Des., **139**(4), p. 041101.
[36] 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.
[37] Falcão, A. F. d. O., 2010, "Wave Energy Utilization: A Review of the Technologies," Renew. Sustain. Energy Rev., **14**(3), pp. 899–918.
[38] Kurniawan, A., Pedersen, E., and Moan, T., 2012, "Modelling of Wave Energy Converters Using Bond Graph," Proceedings of the10th International Conference on Bond Graph Modeling and Simulation, ICBGM'12, Part of SummerSim 2012 Multiconference, Genoa, Italy, July 8–11, pp. 387–394.
[39] 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.
[40] 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, Washington DC, April 10–11, pp. 1–8.
[41] McComb, C., 2018, "Towards the Rapid Design of Engineered Systems Through Deep Neural Networks," Proc. Design Computing and Cognition, Milano, Italy, July 2–4, pp. 3–20.
[42] 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**(Dec.), pp. 23–33.
[43] Vale, C. A. W., and Shea, K., 2003, "A Machine Learning-Based Approach to Accelerating Computational Design Synthesis," Proceedings of the 8th International Conference on Engineering and Design, Stockholm, Sweden, Aug. 19–21, pp. 183–184.
[44] McComb, C., Cagan, J., and Kotovsky, K., 2017, "Capturing Human Sequence-Learning Abilities in Configuration Design Tasks Through Markov Chains," ASME J. Mech. Des., **139**(9), p. 091101.
[45] Puentes, L., Mccomb, C., and Cagan, J., 2018, "A Two-Tiered Grammatical Approach for Agent-Based Computational Design," Proceedings of the ASME 2018 International Design Engineering Technical Conferences and Computers in Engineering Conferences, Quebec City, Quebec, Canada, Aug. 26–29, pp. 1–11.