Data-Driven Heuristic Induction From Human Design Behavior

Lucas Puentes
Department of Mechanical Engineering,
The Pennsylvania State University,
University Park, PA 16802
e-mails: lucaspuentes22@gmail.com;
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

Through experience, designers develop guiding principles, or heuristics, to aid decision-making in familiar design domains. Generalized versions of common design heuristics have been identified across multiple domains and applied by novices to design problems. Previous work leveraged a sample of these common heuristics to assist in an agent-based design process, which typically lacks heuristics. These predefined heuristics were translated into sequences of specifically applied design changes that followed the theme of the heuristic. To overcome the upfront burden, need for human interpretation, and lack of generality of this manual process, this paper presents a methodology that induces frequent heuristic sequences from an existing timeseries design change dataset. Individual induced sequences are then algorithmically grouped based on similarity to form groups that each represent a shared general heuristic. The heuristic induction methodology is applied to data from two human design studies in different design domains. The first dataset, collected from a truss design task, finds a highly similar set of general heuristics used by human designers to that which was hand-selected for the previous computational agent study. The second dataset, collected from a cooling system design problem, demonstrates further applicability and generality of the heuristic induction process. Through this heuristic induction technique, designers working in a specified domain can learn from others' prior problem-solving strategies and use these strategies in their own future design problems.
[DOI: 10.1115/1.4048425]

Keywords: computational synthesis, data-driven engineering

1 Introduction

When completing design tasks, humans can look to their past experiences to plan a strategic course of action for accomplishing their goals, referred to as heuristics [1]. These heuristics can directly guide sequential design changes that execute an abstract design strategy, which permits designers to effectively leap to and explore distant regions of a design space in a manner much more calculated than trial-and-error [2]. However, identifying heuristics and the correct time for their use requires extensive domain experience and familiarity, often leaving less-experienced designers unable to build effective heuristic sets [2,3]. Less-experienced designers can address this issue by implementing generalized heuristics discovered and used by others [4–8]. These generalized heuristics may be very abstract in definition and applied in multiple different ways across domains.

The application of a heuristic depends on the method through which a designer is representing and altering their design. Design grammars are one tool used by designers to concisely represent and navigate design spaces and have been used in prior research to test various generative design search algorithms [9–13]. A grammar ruleset captures the list of all the valid design changes that can be made to a design. In the current study, the grammar ruleset establishes the finite number of different design actions that can be used to transform a design from one state to the next. Puentes et al. [14] explored an approach to grammar-based computational design by implementing a method for heuristic-guided design change sequences, demonstrating that heuristics may provide a boost to early-stage design improvements. Other studies have explored both methods for identifying and inducing grammars [15,16] and methods for inducing heuristics [7,17,18]. However, these studies did not include an automated process for inducing the execution of these heuristics as sequential design actions.

While the previous work by Puentes et al. [14] demonstrated the benefits of using heuristics in grammar-based computational design, the heuristics used in that study were selected based on the researchers' domain experience and predefined to apply within that domain. The execution of these heuristics was specifically programmed, directly translating to a set of design action sequences used to perform each heuristic. Across various domains, the selection of which heuristics to incorporate into design exploration and how to execute them depend on the human designer's own domain experience. To reduce this requirement of domain experience, a process for determining the heuristics used by designers for previous problems may help recommend heuristics for future problems in the same domain. Other works have presented methods for identifying common design heuristics, but they did not include a process for inducing the execution method for these heuristics. This paper seeks to fill this gap with a method for inducing the execution of heuristics as statistically derived design change sequences from human-performed design tasks.

2 Methodology

The proposed method induces the execution of heuristics as sequential design actions by first receiving an iteration-by-iteration timeseries design change dataset as an input and determining significantly occurring sequences of design change actions. Once all significant sequences are extracted, similar sequences are grouped together, denoting that all sequences in a group follow a similarly induced heuristic. An overview of the process is provided in Fig. 1 and is further explained in the following subsections.

2.1 Extraction of Frequent Design Change Action Sequences. To apply the induction method, a timeseries design change dataset must be organized sequentially and clearly define a single design action performed at each design iteration representing the transformation of the design (designated by the first step in Fig. 1). All discrete actions must then be designated by an integer value denoting that specific action (e.g., a truss design action "Add member" may be denoted as "Action 1"), much like how a design change is designated a label within a grammar ruleset. All design repetitions, or full start-to-finish design cycles, contained in the dataset must be represented as one single series of these numerically denoted design changes.

The heuristic induction method begins by analyzing the frequency at which sequences of numerically denoted design actions

Manuscript received February 26, 2020; final manuscript received August 23, 2020; published online October 14, 2020. Assoc. Editor: Joshua Summers.

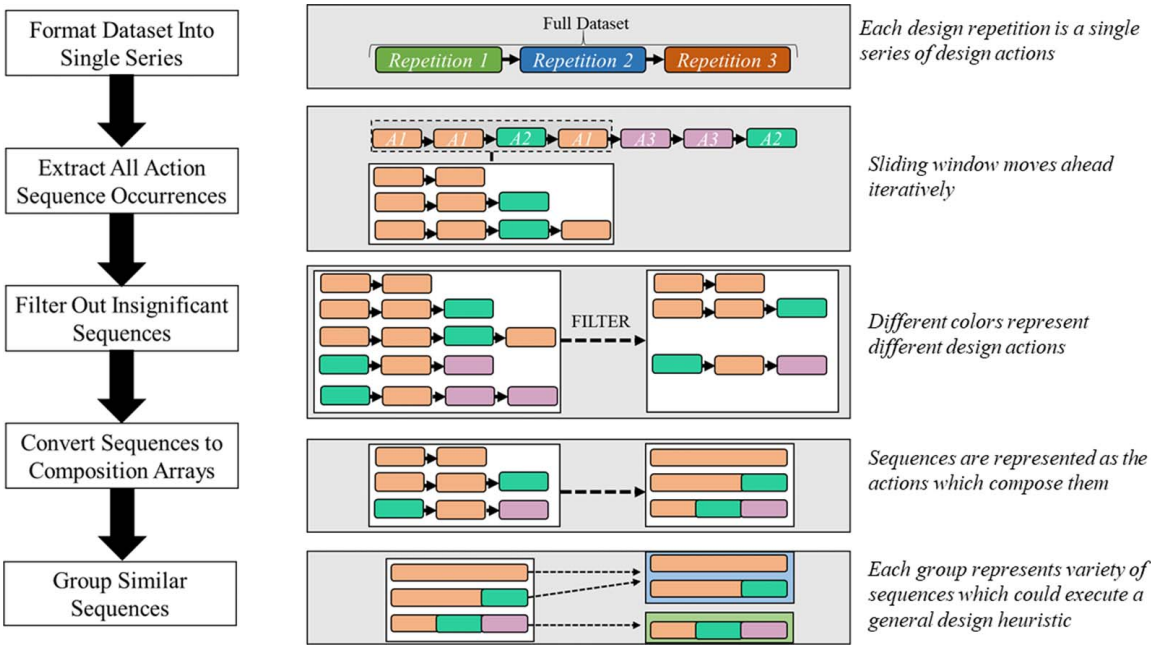
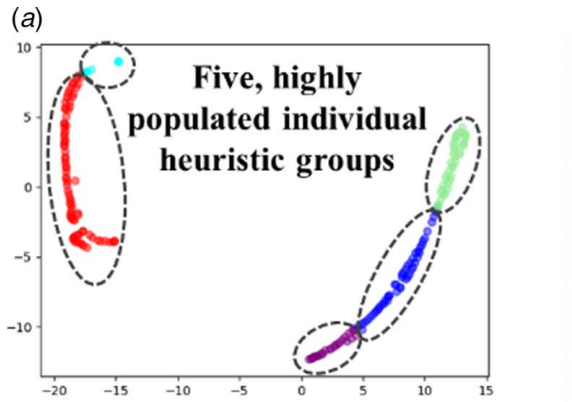
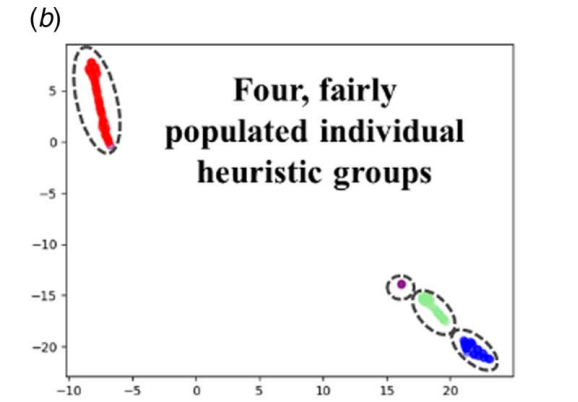
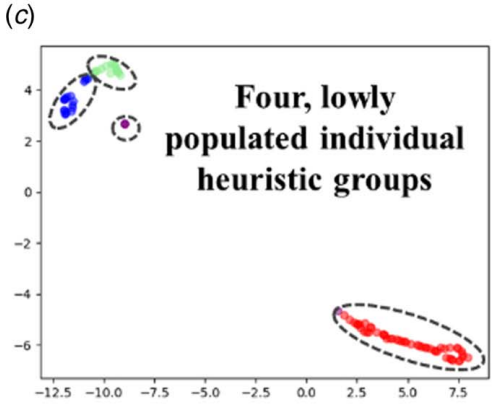


Fig. 1 Flowchart overview of automated design heuristic induction process

occur within the full timeseries dataset (second step in Fig. 2). A sequence is defined as the consecutive design actions that begin at a specified design iteration and end n iterations after. A maximum induced sequence length, L_{max}, must be established at the discretion of the human designer to determine how many design iterations ahead the process will use for individual sequence identification at each iteration. As the statistical analysis performed in this induction methodology only compares sequences of the same length, L_{max} will only affect the induced heuristic diversity that results at the end of this methodology. This diversity refers to the







(d)

	Z=1.0	Z=1.5	Z=2.0
Increase complexity	✓	✓	✓
Increase scale	✓	✓	✓
Decrease scale	✓	✓	✓
Standardize components	✓	✓	✓
Increase complexity then decrease scale	✓	-	-

Fig. 2 t-SNE representations of induced heuristic groups from truss design dataset for (*a*) z-score of 1.0, (*b*) z-score of 1.5, and (*c*) z-score of 2.0. Maximum sequence length for all three cutoff values is 15. (*d*) Generalized heuristics identified for each z-score cutoff.

number of sequence variations that may represent a single heuristic. Starting at the first design iteration, a sliding window approach identifies all sequences that occur in the dataset. The window begins on the current design iteration and ends at the iteration that is L_{max} iterations ahead of the current. Within this window, sequences of varying length are identified, where sequences of length n begin with the design change at the current iteration and end at the iteration $n-1$ steps ahead. The window then slides ahead one iteration and the process repeats while counting the occurrence of every discrete identified sequence.

The identified sequences that occur the most frequently throughout the dataset are determined through statistical analysis and extracted (third step in Fig. 1). This statistical analysis compares the occurrence rate of all discrete sequences within each individual sequence length. This ensures that sequences of a certain length are not compared to sequences of another length, as very short sequences are by nature much more likely to occur and would thus make the occurrence rate of very long sequences seem insignificant. The occurrence rate is defined as the number of occurrences of a specific sequence divided by the total number of sequence occurrences of that same length. For each individual sequence length, the mean and standard deviation of sequence occurrence rates are calculated. Using these statistical metrics, a z-score calculation is made for all identified sequences. High z-scores correlate to statically high occurrence sequences. A z-score cutoff is determined by the designer in order to filter out sequences deemed as insignificant. This cutoff is largely dependent on the designer's preference for induced heuristic diversity, with lower cutoffs leading to more diverse heuristic sets.

2.2 Unsupervised Grouping of Similar Extracted Sequences. Individual significantly occurring sequences are grouped together based on the similarity of their design change composition, corresponding to the multiple execution variations of a single generalized heuristic. To determine sequence similarity, all sequences must be converted to an array containing the composition of all design actions within that sequence (fourth step in Fig. 1). For example, in a dataset with four design actions, the sequence *{Action 1, Action 1, Action 1, Action 4}* is converted to the action composition array $\boldsymbol{A} = [0.75, 0.0, 0.0, 0.25]$ by dividing each action occurrence by the total number of actions. Pairwise distances between each extracted sequence and all other extracted sequences are calculated using distance Eq. (1),

$$d = \sum_{i=1}^{i_{max}} |\boldsymbol{A}_{1_i} - \boldsymbol{A}_{2_i}| \tag{1}$$

where the similarity distance, d, is calculated as the sum of the absolute element-wise differences of the action composition of the first sequence, $\boldsymbol{A}_1$, and the action composition of the second sequence, $\boldsymbol{A}_2$, for all array elements, $i = 1$ to i_{max}.

This paper uses a mean shift clustering algorithm to group the extracted sequences (final step in Fig. 1). This algorithm seeks to locate the optimal number and locations for centroids within a dataset that are a mean of the data points in a specific surrounding region [s]. The kernel bandwidth values used in this paper are determined by a bandwidth estimation algorithm that performs a nearest-neighbor analysis on the full dataset [20]. Though other clustering algorithms exist and could be used instead, the contributions of this paper are not dependent on this algorithm selection. Each resulting grouping of sequences represents a generalized design heuristic that may be executed through one of the sequences in that group. Human designers are currently required to interpret which generalized heuristics are represented by induced groups.

2.3 Design Dataset Applications. The proposed methodology is applied to the sequential design action datasets of two separate human design studies: truss configuration design and home cooling system layout [21–24]. For the truss design study [21,23], 16 teams of three senior-level university students were tasked with constructing truss configurations with minimal mass and a factor-of-safety of over 1.0. For the home cooling system layout study [22,24], 68 senior-level and graduate-level university students were assigned to work as individuals or in three-person teams and tasked with designing an Internet-connected home cooling system to minimize house temperature and system cost. Participants in both studies completed the design task by iteratively selecting a design action and applying it using the provided computer-based interface. The number of discrete component-level design actions available for each study was nine. The total number of actions permitted for the full design process was unlimited for the truss design but limited to 50 for the cooling system layout.

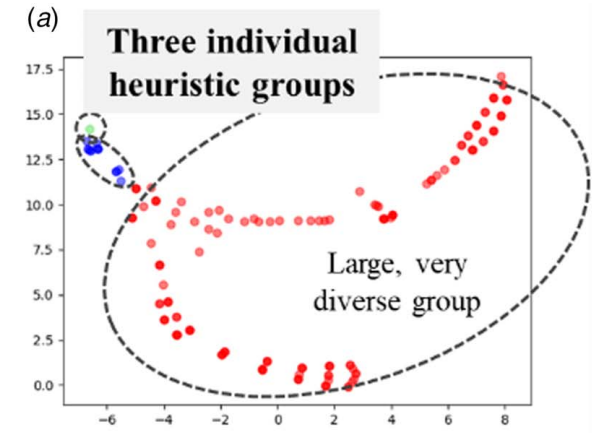
3 Results and Discussion

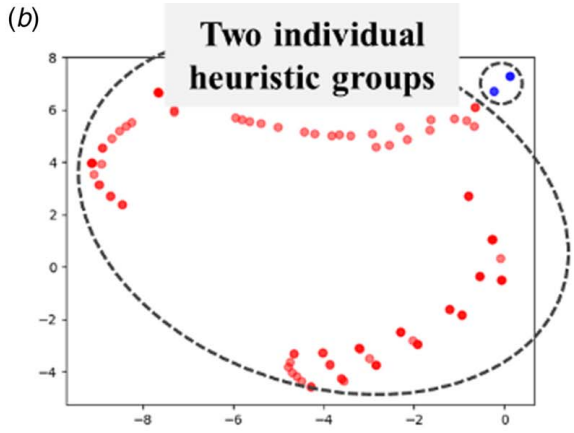
The following subsections present the findings of the application of the heuristic induction method to the two design study datasets using z-score cutoff values of 1.0, 1.5, and 2.0.

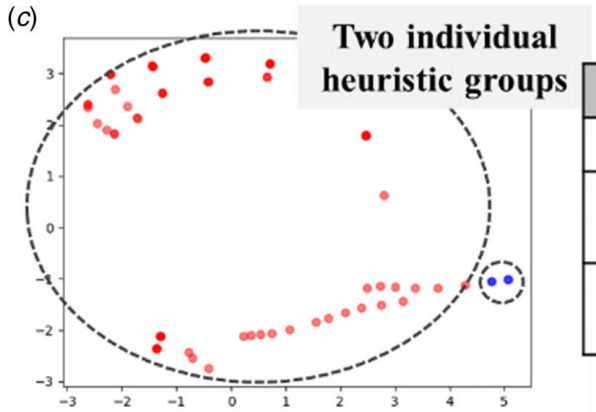
3.1 Truss Design Study. For the truss design dataset application, the tested L_{max} is 15 to be comparable to the average expected heuristic length to the heuristics used by Puentes et al. [25]. Five heuristic sequence groups are identified for z-scores of 1.0 and 1.5, while four groups are identified for a z-score of 2.0. Closer inspection confirms that the groupings of sequences are consistent across z-score values and the diversity of each group decreases as the z-score cutoff increases. A comparison of the induced heuristic sequences for each z-score is found in Fig. 2. As individual sequences are represented in their action composition arrays, a high-dimensional space, a T-distributed Stochastic Neighbor Embedding (t-SNE) embedding algorithm is used to visually represent the sequences as points in a two-dimensional space [26], but only after clustering is performed. The t-SNE algorithm uses the pairwise similarity distances, d, of all sequences to calculate this embedding. Highly similar sequences are visually close, and unsimilar sequences visually far. It is important to note that the visual distances are not the same as the similarity distances used to create the sequence group assignments. The results shown in Figs. 2 and 3 (see Sec. 3.2) used a t-SNE perplexity value of 30, though the visual representation is stable for perplexity values from 10 to 45.

The interpretation of the generalized heuristics portrayed in each group is determined by the researchers based on prior domain experience. At the lower z-score cutoffs, four of the groups suggest the execution of straightforward heuristic principles for increasing design complexity, increasing design scale, decreasing design scale, and standardizing components. In this domain, executing these principles would result in sets of design action sequences that are consistent with those found in each of these four induced groups. The final group contains sequences that suggest the execution of a higher-level, two-step heuristic that first increases design complexity and then decreases the scale. In this domain, the sequential implementation of these two heuristics would result in a set of design action sequences that resemble those found in this final induced group. This multi-step heuristic demonstrates a strategic problem-solving approach in which human designers understand the expected outcome of two individual heuristics, and then combine them in a way that capitalizes on these expected outcomes.

The set of generalized design heuristics used previously by Puentes et al. in Ref. [14], and later adapted for Ref. [25], was predefined and specifically programmed for use by automated computational design agents. The induced heuristics of the current paper are compared to the latter study, as it used the same human truss design dataset and design actions. The ideal set of generalized design heuristics to use in this domain was unknown to the human designers who established this predefined set. In this truss design dataset, four of the five induced heuristic groups resemble heuristics used in the prior agent-based study, indicating that the set chosen is overall effective for implementing heuristics which







(d)

	Z=1.0	Z=1.5	Z=2.0
Increase complexity	✓	✓	✓
Tune multiple coolers	✓	✓	✓
Reorganize multiple coolers	✓		

Fig. 3 t-SNE representations of induced heuristic groups from cooling system layout design dataset for (*a*) z-score of 1.0, (*b*) z-score of 1.5, and (c) z-score of 2.0. Maximum sequence length for all three cutoff values is 10. (*d*) Generalized heuristics identified for each z-score cutoff.

are common in the truss domain. The fifth, multi-step induced heuristic was not previously considered for use in the predefined set. Meanwhile, the final prior predefined heuristic does not seem to be a frequent heuristic used in the truss domain, suggesting that the predefined set could have been more efficient in the previous study by not trying to apply a less effective heuristic.

3.2 Home Cooling System Layout Study. For the cooling system layout dataset application, the tested L_{max} is 10, due to the 50-iteration limit placed on student designers. Three heuristic groups are identified for z-scores of 1.0 and 1.5, while only two are identified for a z-score of 2.0. As with the truss design dataset, the interpretation of induced heuristics is determined by the researchers. The first induced heuristic group contains sequences that focus on adding multiple new design components. The second group, which is not present at higher z-score cutoffs, reorganizes the spatial arrangements of multiple coolers. Rather than a single apparent generalized heuristic, the third group contains sequences indicating three distinct subgroups that begin with different design actions (tuning existing coolers, adding new coolers, or rearranging existing coolers) but share a commonality in that they conclude with multiple *tune cooler* actions. A t-SNE-based visual comparison of the induced heuristic sequences for each z-score is found in Fig. 3.

The findings of this application correlate well with the findings of McComb et al. [27], in which Markov chains were applied to the same cooling system layout dataset as this current paper. A Markov chain is a mathematical model that determines the probability of transitioning from one state to another in a finite state system [28]. In the context of this dataset, Markov chains determined the likelihood of performing one design action after another has been performed. McComb et al. found that statistically, the most occurring transition was consecutively performing the *tune cooler* action, which resembles the induced heuristic group containing mostly sequences with a repeating *tune cooler* action. Additionally, this correlation may provide a reason for the presence of three distinct subgroups within that single group. Slight variations of a repetitive *tune cooler* sequence would be grouped within the same sequence set even as these variations continue to differ slightly. Other frequently occurring transitions identified by Markov chains were (a) adding a processor then adding a sensor and (b) adding a sensor then adding a cooler, which resembles the action composition of a large portion of the sequences within the induced heuristic group for adding more components. Lastly, the transition for moving a cooler and then moving another cooler occurred less often than the other frequently occurring transitions but provides insight as to why the induced heuristic group for reorganizing coolers is identified at only lower z-score values.

4 Conclusions

This paper introduces a methodology for inducing the execution of heuristic strategies in a design domain from existing design problem datasets. This heuristic induction method is applied to two separate human design study datasets: truss design and home cooling system layout. Through automated heuristic induction, designers

can identify and understand which design heuristics may be applicable in a domain and the forms in which they may be implemented.

The findings of this study are limited in scope by the level of domain expertise likely possessed by the student designers who completed the original design studies to produce the used datasets. Though the induced heuristic sequences may not directly reflect those used by experts in these domains, the purpose of this work is to demonstrate the ability of the presented method to extract frequently occurring design change sequences from a dataset and group them together based on similarity. To that end, this student data are sufficient.

Future improvements to the heuristic induction method will seek to address its limitations. Currently, the method requires the designer to personally select a maximum sequence length and sequence significance cutoff. However, this selection is critical to the resulting heuristic groups. Future work will explore ways to standardize the selection of these values and their relation to resulting induced heuristic diversity. These improvements will help further validate the findings of this study. Another area of improvement is in the automated interpretation of induced heuristic groups, which would allow for the automated transfer and translation of these heuristics to other design problems or domains.

Acknowledgment

This material is based upon work supported by the Defense Advanced Research Projects Agency through cooperative agreement N66001-17-4064.

Conflict of Interest

There are no conflicts of interest.

Data Availability Statement

The data and information that support the findings of this article are freely available at 10.1016/j.dib.2018.02.078 and 10.1016/j.dib.2017.08.050 The data and information that support the findings of this article are freely available at 10.1016/j.dib.2018.02.078 and 10.1016/j.dib.2017.08.050 The data and information that support the findings of this article are freely available at 10.1016/j.dib.2018.02.078 and 10.1016/j.dib.2017.08.050

References

[1] Nisbett, R. E., and Ross, L., 1980, *Human Inference: Strategies and Shortcomings of Social Judgment*, Prentice-Hall, Englewood Cliffs, NJ.
[2] 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.
[3] Cross, N., 2004, "Expertise in Design: An Overview," Des. Stud., **25**(5), pp. 427–441.
[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., 2014, "A Case-Study Analysis of Design Heuristics in an Upper-Level Cross-Disciplinary Design Course," 2014 ASEE Annual Conference & Exposition, Indianapolis, IN, June 15–18, pp. 24.23.1–24.23.17.
[6] 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.
[7] Yilmaz, S., Seifert, C., Daly, S. R., and Gonzalez, R., 2016, "Design Heuristics in Innovative Products," ASME J. Mech. Des., **138**(7), p. 071102.
[8] Blösch-Paidosh, A., and Shea, K., 2019, "Design Heuristics for Additive Manufacturing Validated Through a User Study 1," ASME J. Mech. Des., **141**(4), p. 041101.
[9] Schmidt, L. C., and Cagan, J., 1997, "GGREADA: A Graph Grammar-Based Machine Design Algorithm," Res. Eng. Des., **9**(4), pp. 195–213.
[10] 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.
[11] Königseder, C., and Shea, K., 2014, "Systematic Rule Analysis of Generative Design Grammars," Artif. Intell. Eng. Des. Anal. Manuf., **28**(03), pp. 227–238.
[12] 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.
[13] Knight, T., 1998, "Designing a Shape Grammar: Problems in Predictability," *Artificial Intelligence in Design '98*, 1, J. S. Gero, and F. Sudweeks, eds., Springer, Dordrecht, pp. 499–516.
[14] Puentes, L., Cagan, J., and Mccomb, C., 2020, "Heuristic-Guided Solution Search Through a Two-Tiered Design Grammar," ASME J. Comput. Inf. Sci. Eng., **20**(1), p. 011008.
[15] Whiting, M. E., Cagan, J., and LeDuc, P., 2018, "Efficient Probabilistic Grammar Induction for Design," Artif. Intell. Eng. Des. Anal. Manuf., **32**(02), pp. 177–188.
[16] Eichhoff, J. R., Baumann, F. W., and Roller, D., 2017, "In Search of Missing Design Rules : Using Rule Induction to Extend Existing Rule Bases," J. Adv. Technol. Eng. Stud., **3**(4), pp. 150–169.
[17] Studer, J. A., Yilmaz, S., Daly, S., and Seifert, C., 2016, "Cognitive Heuristics in Defining Engineering Design Problems," Proceedings of the ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, ASME, Charotte, NC, pp. 1–11.
[18] 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**(4), p. 041008.
[19] Fukunaga, K., and Hostetler, L. D., 1975, "The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition," IEEE Trans. Inf. Theory, **21**(1), pp. 32–40.
[20] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E., 2011, "Scikit-Learn: Machine Learning in Python," J. Mach. Learn. Res., **12**(85), pp. 2825–2830.
[21] McComb, C., Cagan, J., and Kotovsky, K., 2015, "Rolling With the Punches: An Examination of Team Performance in a Design Task Subject to Drastic Changes," Des. Stud., **36**(C), pp. 99–121.
[22] 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.
[23] McComb, C., Cagan, J., and Kotovsky, K., 2018, "Data on the Design of Truss Structures by Teams of Engineering Students," Data Br., **18**, pp. 160–163.
[24] Mccomb, C., Cagan, J., and Kotovsky, K., 2017, "Data on the Configuration Design of Internet-Connected Home Cooling Systems by Engineering Students," Data Br., **14**, pp. 773–776.
[25] Puentes, L., Raina, A., Cagan, J., and McComb, C., 2020, "Modeling A Strategic Human Engineering Design Process: Human-Inspired Heuristic Guidance Through Learned Visual Design Agents," International Design Conference Design 2020, Dubrovnik, Croatia, pp. 1–10.
[26] Maaten, L., Van Der, and Hinton, G., 2008, "Visualizing Data Using T-SNE," J. Mach. Learn. Res., **9**, pp. 2579–2605.
[27] 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.
[28] Strook, D. W., 2005, *An Introduction to Markov Processes*, Springer Science and Business Media, Berlin.