Res Eng Design
DOI 10.1007/s00163-015-0190-3

ORIGINAL PAPER

Synergistic human-agent methods for deriving effective search strategies: the case of nanoscale design

Paul Egan · Jonathan Cagan · Christian Schunn · Philip LeDuc

Received: 15 March 2014 / Revised: 16 October 2014 / Accepted: 13 February 2015

Abstract Complex systems are challenging to understand and design, and even more so when considering nanoscale reasoning. This paper introduces a synergistic cognitive and agent-based methodology for deriving effective strategies for human searches in optimization design tasks. The method consists of conducting cognitive studies to determine effective human search approaches, rapidly testing algorithmic variations of human strategies using software agent automation, and finally providing the highly effective agent-refined strategies to humans. The methodology was implemented by developing a graphical user interface (GUI) of myosin biomotors and conducting a baseline cognitive study to determine how users effectively search for optimal biosystem designs. The best human designers typically searched local to their current best solution, utilized univariate searches, and may have learned and applied parametric knowledge. These trends informed rule-based agent strategies, and testing variations of rules resulted in the discovery of highly effective strategies using initial random searches, univariate searches to learn parameter relationships, and greedy local searches to apply knowledge. The GUI was modified to aid users in implementing two of the highest performing agent strategies in a final cognitive study. These users provided with the agent-refined strategy performed better than users with no provided strategy during the baseline cognitive study. When agents and users were provided myosin domain knowledge prior to searching, convergence on high-quality designs occurred earlier, which suggests that even experts in the domain could benefit from the agent-derived strategies. These findings demonstrate the power of synergistic human- and agent-based approaches, in which cognitive-based findings can reveal strategies that are refined by agents that generate search strategies for greatly improved user performance. The synergistic methodology extends beyond nano-based applications and could generally aid designers in discovering effective decision-making approaches across a broad range of domains.

Keywords Agents · Cognitive · Optimization · Biology · Nano · Complex

1 Introduction

Complex systems present challenges for engineering design in many domains of application (Ottino 2004), especially when considering the difficulties human users have in understanding and applying knowledge concerning these systems (Chi et al. 2012; Hmelo-Silver et al. 2007). In this paper, our goal was to develop a method for finding effective strategies for human traversal of complex system design spaces. This method consists of using graphical user interfaces (GUIs) to facilitate synergistic cognitive (Kuhn et al. 2008) and agent-based (Landry and Cagan 2011) approaches of discovering and refining strategies for searching complex system design spaces. This synergistic human-agent approach is promising because cognitive studies can provide a starting point for identifying how users initially approach complex system design, and then software agents can rapidly test variations of strategies inspired by successful user searches to find more effective search approaches.

Complex systems are particularly challenging to design because (1) complex systems consist of many interacting parts across multiple scales of space and time and (2)

P. Egan · J. Cagan (✉) · P. LeDuc
Carnegie Mellon University, Pittsburgh, PA, USA
e-mail: cagan@cmu.edu

C. Schunn
University of Pittsburgh, Pittsburgh, PA, USA

Published online: 06 March 2015

complex systems often exhibit counterintuitive emergent behavior (Guckenheimer and Ottino 2008). One domain that exemplifies both of these criteria is nanoscale biomechanical system design (Egan et al. 2013a), such as the design of synthetic muscle tissue (Bach et al. 2004) and nano-actuators (Neiman and Varghese 2011) that incorporate myosin biomotors (Fig. 1). Design of these myosin-based systems is particularly challenging because there are many design variables that influence emergent system performance, with changes to each nanoscale design variable having multiple causal effects on nano- and system-level performance that are often nonlinear (Egan et al. 2013a).

Myosin systems consist of individual myosin proteins that stochastically attach to actin filaments and exert propulsive force (a video describing how myosins work with a supporting animation is available at http://youtu.be/kLX-0Zdizk8). Each myosin utilizes the energy of one adenosine triphosphate (ATP) molecule per attachment cycle. Myosin systems are particularly counterintuitive because individual myosin mechanics and chemistry are coupled with the behavior of the system as a whole. In muscles, for example, as tension grows and increases the load on each myosin, the myosins handle the greater load while utilizing less energy because they interact with actin filaments at a lower rate. Although removing myosins from a system will reduce its energy consumption, which generally leads to greater efficiency, a minimum energy threshold must be satisfied or the system will dissociate and completely cease functioning (Harada et al. 1990). Many parameters in myosin design are coupled to the same outputs (increases in myosin lever arm length, attachment rate, and detachment rate can all alter ATP use and required energy thresholds nonlinearly), which suggests these systems could be difficult for designers to search effectively, even when only a few parameters are considered.

Our goal was to aid designers in handling these non-obvious coupled relationships by facilitating their ability to effectively manipulate design variables when configuring complex systems. Our first step was to measure human search effectiveness by developing a GUI that tracks human searches of myosin design spaces that are presented as nonlinear programming (NLP) optimization problems. NLP tasks are well suited for the myosin problems because they enable a simple representation of constraints and an objective function that enables humans to search an otherwise complicated optimization problem. Additionally, they are solvable by many traditional algorithmic engineering search approaches (Belegundu and Chandrupatla 2011). When these tasks are solved by a user, the user is expected to search the design space for a configuration that best fulfills a design goal without violating any constraints, such as increasing the filament velocity of a system without exceeding a maximum amount of ATP utilized by the system. These considerations make each NLP task meaningful and tied to real-world issues and trade-offs. The NLP representation approach is also well suited for cognitive studies because it enables tuning the difficulties of these problems through adding multiple constraints to a problem statement, or changing the number of relevant outputs to consider.

An essential aspect of our approach is proposing and identifying effective user search strategies, which can be informed by engineering optimization approaches and cognitive studies. For instance, in cognitive investigations of human design, it has been suggested that humans are limited by their cognitive capacity, meaning search decisions should require low effort (Hirschi and Frey 2002) and could be facilitated by simply making small changes to an existing design. Such an approach is also supported from an engineering standpoint, because many search strategies use information based on the current best known solution, such as extended pattern search (Aladahalli et al. 2007). Similarly, univariate decision-making processes (Chen and Klahr 1999) are often effective for humans because they facilitate scientific reasoning for learning how inputs affect outputs in isolation, while also being beneficial from an engineering perspective because they reduce the effects of parameter coupling on searches. A final cognitive-based approach could leverage human learning of parametric relationships using short-term memory (Hirschi and Frey 2002), thus allowing users to apply that knowledge toward a better solution from an engineering perspective. These proposed strategies inspired by cognitive and engineering search approaches are summarized in Table 1. They represent only a small subset of the many strategies our synergistic human-agent approach could investigate, but serve as a starting point for testing the methodology in this paper.

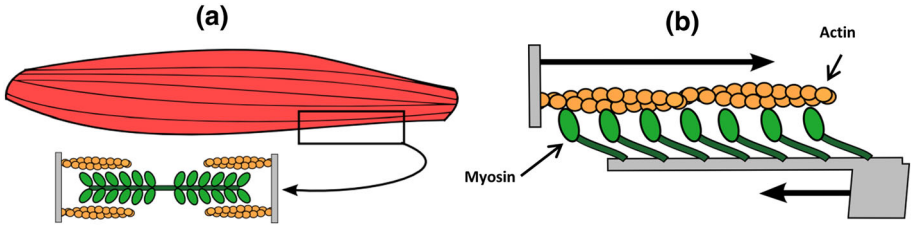

Fig. 1 Schematics of **a** synthetic muscle and **b** nano-actuator technologies. An additional explanation and animation of myosins presented via the GUI are available at http://youtu.be/kLX-0Zdizk8

Table 1 Cognitive-based search approaches

Proposed cognitive-based search approaches			
Search approach	Human reasoning process	Cognitive basis	Engineering basis
Compare to Best	New designs are configured that are similar to the previously best known design	Low cognitive load	Incremental improvements in design quality
Univariate	One design variable is manipulated at a time	Facilitates scientific reasoning	Mitigates effects of parameter couplings
Learn and Apply	Parametric relationships learned during the search process are applied toward finding better designs	Correlations are stored in short-term memory	Directs search process toward an optimum

Our approach in this paper for developing and testing a human-agent synergistic methodology is to first have many human designers search timed design tasks with the GUI. These searches are then categorized according to user search success, and the trends are identified using Table 1 approaches as a starting point to discover the best human search approaches. These best human approaches are then used to inform strategies automated by computational agents that solve the same tasks as users. Once varied strategies are attempted by agents, their best found strategies are provided to human designers. Agents are useful at this step because they can explore the benefits of each strategy faster and more consistently than is accomplishable with further cognitive studies. A symbiotic relationship among human designers and computational agents emerges in this approach, as the initial trends from user results plant the seeds for deriving strategies that are implemented by agents. When the most successful agent strategy is returned to users, it is a highly effective algorithmic strategy that remains consistent with how the highest performing users initially approached the problem. These highly effective strategies refined through synergistic human-agent approaches should prove extendible to other complex systems, and the synergistic methodology as a whole could help in discovering helpful human decision-making strategies in a wide variety of applications.

2 Background

Relevant background is presented, which informs the development and testing of our synergistic human-agent methodology and includes an overview of cognitive processes in complex system design, GUIs, engineering optimization, and agent-based design approaches.

2.1 Cognitive processes in complex system design

Cognitive studies have demonstrated that it is difficult for people to understand the causal mechanisms in complex systems (Hmelo-Silver et al. 2007), and that experts reason about complex system interactions differently than novices. Such considerations of novices and experts have been investigated in engineering contexts and demonstrated that novices can perform similar to experts with minimal training before problem solving (Wolf et al. 2011), which suggests it would be possible to provide novice users in our studies with information that enables them to simulate experts in the domain. Learning complex systems has been investigated with software tools for manipulating natural biological systems that operated primarily at the macro-scale, which was demonstrated to improve user understanding (Vattam et al. 2011). These findings suggest that it is possible for users to learn while using software tools, which can enable them to learn and apply information during the problem-solving process, and that learning is even possible in unfamiliar biological domains.

It is also important to aid users in forming effective decisions when changing and evaluating designs. Many cognitive studies have looked into the most beneficial way for users to search for solutions in quantitative problems (Chen and Klahr 1999) and found that univariate approaches to manipulating inputs are useful in developing scientific reasoning skills and could extend to design. Such techniques are useful for informing strategies for human and agent use. Past studies have shown that users can have difficulties in solving parametric design problems when as little as three design variables are coupled to performance outputs (Hirschi and Frey 2002), which suggests that humans may benefit from aid in even simple myosin design tasks. These findings provide a basis for presenting design tasks to users that only have a few design inputs and outputs, and to ensure that each input can be changed independently of one another, such that users may utilize univariate searches.

2.2 Graphical user interfaces for design

A variety of tools exist for modeling molecular scale natural and synthetic biological systems, with a few examples

being TinkerCell (Chandra et al. 2009) for considering components inside cells, an agent-based biochemical reaction simulator NFSim (Michael et al. 2011), and Gene designer (Villalobos et al. 2006), which is a visual design tool for de novo gene configurations. These tools demonstrate the feasibility of building user interfaces for designing biological systems, but focus heavily on biochemical approaches, whereas myosin systems are mechanochemical and may require new representation approaches. The multiple parameters and causal influences in the myosin domain suggest the need for multi-dimensional data visualization techniques (Zhang et al. 2012). An effective approach found by a past engineering study simultaneously presented many relevant inputs and outputs for designs across a series of plots (Wolf et al. 2011), which is a well-suited approach for representing the multiple inputs/outputs in myosin NLP optimization tasks. It is also possible to build a GUI that enables its extension to many different systems, such as past GUIs that have been beneficial for I-beam, desk lamp, aircraft wing, and manufacturing system designs with minor modifications (Simpson et al. 2007b).

Another important consideration is that the GUI facilitates human approaches to searching design spaces, with two concerns being the timeliness of feedback provided and also how users manipulate designs. A research study on multimedia tools in virtual reality demonstrated the negative effects of information lags in interactive simulations (Viciana-Abad et al. 2011). Feedback delays have also been studied in an engineering context for designing multiple systems, with one example being I-beam optimization (Simpson et al. 2007a). The addition of real-time feedback generally improves user interaction with a GUI and can be used as a basis for assessing the helpfulness of information provided by the GUI. The manipulation of feedback delays simulates design scenarios in which rapid outcome information is not readily available to a designer (Pretz 2008), such as in performing wet-lab experiments or computationally expensive simulations. Such expensive simulations have been explored in the past specifically for designing in the myosin domain (Egan et al. 2013a) and are capable of promoting insights that are not obtainable through faster evaluation methods. Thus, initially exploring such scenarios of delayed feedback can validate that a GUI is providing information that is useful for the design process.

2.3 Engineering optimization

There are many different approaches to searching design spaces (Belegundu and Chandrupatla 2011), and the most well-suited approach to use depends on the type of space being searched. Search strategies for complex systems can range from deterministic approaches that tend to focus on finding local solutions to stochastic approaches that widely search a design space (DuPont and Cagan 2012). Therefore, in large search spaces with many local optima, an algorithm with stochastic properties is essential. However, the stochastic algorithms come at a cost of being more computationally demanding in terms of function evaluation count because they search a broader range of candidate designs. In complex search spaces where the global optimum is highly difficult to isolate, heuristic approaches are often used to find high-performing designs; a recent adaptive stochastic approach consisted of software agents using genetic search algorithm heuristics to effectively traverse a large search space (Landry and Cagan 2011). Hybrid approaches are also highly effective and can utilize stochastic searches to explore a broad portion of a design space, while utilizing deterministic searches to pinpoint optimal designs once candidate starting locations are found, which is similar to the approach utilized in simulated annealing (Shea et al. 1997).

Another successful hybrid approach that combines stochastic and deterministic searches is the extended pattern search (Aladahalli et al. 2007), which has recently been deployed for searching complex design spaces for optimal wind farm layouts (DuPont and Cagan 2012). In the extended pattern search, the search space of a layout problem is traversed by first stochastically placing components in particular locations. Component locations are then optimized through deterministic local searches by conducting univariate manipulations of component positions and only keeping moves that result in global improvement. To avoid the early convergence on low-performing optima caused by reliance on deterministic position adjustments, additional iterations are conducted through stochastically popping the locations of some components to a new space and applying the deterministic placement again. Through iterations of deterministic and stochastic search processes, the algorithm maintains a high degree of exploration while also converging on high-quality designs. Such findings suggest that hybrid strategies are a powerful approach that could be utilized in human-agent searches, and that univariate changes to designs are effective even in complex design scenarios.

2.4 Agent-based computational design

Software agents are computational objects that have varied capabilities in perceiving and manipulating virtual environments, storing information, and automating the process of searching a design space using rule-based approaches. Past software agent approaches in engineering design have included implementations with agents mimicking design teams (Olson et al. 2009), agents evolving in a manner similar to genetic algorithms (Hanna and Cagan 2009), and

expert agents providing suggestions as they learn user preferences (Schiaffino and Amandi 2009). In these applications, agents are independent and often autonomous computational objects that sense information from their environment in the form of assessing design representations. Through using programmed rules and stored memory, they make effective decisions toward reaching a design goal.

In the A-design approach, populations of agents each have their own independent design goals such as configuring, sorting, or diversifying designs. Through an iterative process, the population of agents searches a design space (Campbell et al. 1999). The A-design agents were also developed to reason about the functional requirements of a design and fulfill its function by creating component configurations in the domain of electromechanical devices. Later implementations utilized cognitive-based engineering approaches to improve agent performance through simulated chunking of information, similar to human experts. These findings suggest that cognitive-based strategies are useful in programming successful agent approaches.

Another successful approach was inspired through evolutionary algorithms, where individual agents possessed varied strategies and the population of agents evolved throughout the solution process to optimize their approach for the problem at hand (Landry and Cagan 2011). In this case, agents had no intelligence and mimicked binary string operations from genetic algorithms. The evolution of the agents aimed to configure an optimal search strategy for deployment during each phase of a search process. Findings from this approach are suggestive of the need to consider many different agent strategies in finding an optimal search approach, and that optimal agent strategies differ based on problem types. Other studies have incorporated learning into computational agent evolution (Buczak et al. 2005), which suggests it is possible to combine cognitive-based learning approaches in refining agent search strategies—a pivotal aspect of our synergistic agent-human methodology.

3 Methods: design tasks, GUI, and pilot study

Before initiating the synergistic human-agent methodology, a process must be developed and tested for retrieving and analyzing human search data. This process consists of generating a set of design tasks, building a GUI for collecting human search data, and then testing it with users and assessing the results.

3.1 Design tasks

Design tasks are NLP optimization problem statements with goals and constraints that represent myosin-based technological requirements. For instance, one task may have a goal of highest filament velocity, in which case a user or agent would be required to manipulate myosin design variables in order to achieve the highest velocity possible. If a problem is constrained, such as having a maximum allowable energy, the user or agent would have to search for a system that performs the best while still meeting the constraint—designs that do not meet constraints are considered invalid. Each design task only has one goal performance output that may or may not be constrained and represents the objective function, and a second performance output that may also be constrained. Through these design task considerations, the GUI was developed to enable users to manipulate myosin design variables and view the relevant outputs of a design task.

3.2 Design modules and their integration to form a scene

Our visual design tool consists of interactive modules that aid the user for NLP optimization tasks with a single objective function goal. These modular tools are integrated to form interactive tutorial and design tasks scenes for a user that are presented sequentially (video demonstration at http://youtu.be/XOi3n-XwAXQ). The software tool was implemented in Java with the JavaFX library supporting the graphics and interface and is presented in Fig. 2, which demonstrates each module of a design scene for a task with four design inputs and two performance outputs.

In the figure, the task description presents information about the design goal and constraints. The design input sliders enable a user to alter values of four independent myosin system design variables and evaluate the current design with a button press (these four myosin design variables are always present regardless of the number of performance outputs). As myosin values are altered, new visualizations of the system are rendered: (1) a myosin's lever arm length is represented by the height of the myosin graphic, (2) a myosin's attachment rate is represented by the size of the "up" arrow to the left of the myosin, (3) a myosin's detachment rate is represented by the size of the "down" arrow to the right of the myosin, and (4) the number of myosins is proportionally reflected by the number of myosins in the system rendering graphic. The maximum values for each input are a myosin lever arm length l of 20 nm, myosin attachment rate k_{on} of 4000 s^{-1}, myosin detachment rate k_{off} of 2000 s^{-1}, and a maximum number of myosins N_{myo} of 100. The lever arm, attachment rate, and detachment rate are represented in a non-dimensionalized quantity to avoid overly confusing units and values for users who are unfamiliar with the system. These variables are weighted from 0.2 to 1.0 of the parameter's maximum value according to the five possible input

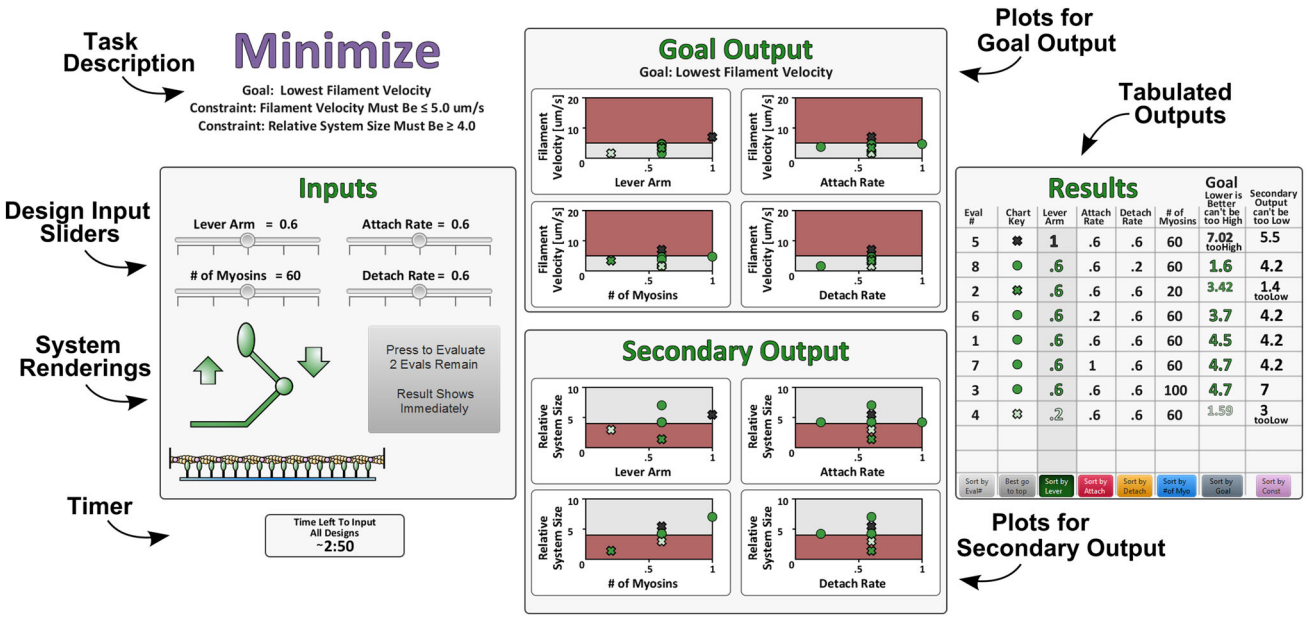

Fig. 2 Labeled screen capture of design scene integrating all software tool modules (color figure online)

possibilities tied to each slider. The sliders were tuned such that a user increasing a slider input linearly resulted in its corresponding myosin design input increasing linearly (e.g., if a slider was at its lowest value of 0.2 and then moved to its value of 0.8, and the corresponding myosin input was lever arm length, it would increase from 4 to 16 nm). Because there are four sliders and five input positions for each slider, 625 design possibilities exist. The resulting design space is quite difficult for users to navigate when considering the counterintuitive parameter interactions present and limited time a user has to search the space during our timed user studies. The mathematical myosin model used in this study has been empirically validated for each of these myosin design variables (Egan et al. 2013a) and is described in section "Myosin mathematical modeling" in Appendix. The chosen ranges for myosin design inputs are informed from empirical studies in the myosin domain (Howard 2001).

The tabulated results in Fig. 2 display the input parameters used for each design evaluation and the outputs for how that design performs once it is evaluated. Each column represents a different design metric and from left to right are "Eval #" that represents the order a design was evaluated, "Chart Key" that displays symbols for the plotted solution, "Lever Arm," "Attach Rate," "Detach Rate," and "# of Myosins" columns that represent their respective design inputs, and the "Goal" column that has values of the objective function and details about the constraint (e.g., "Higher is Better, can't be too High" in Fig. 2). If there is a second output parameter, it appears in the column to the right of the "Goal" column. Each row represents a single design evaluation with the exception of the top row being a header and the bottom row housing sort buttons. Sort buttons always move higher value designs with respect to the sorted parameter of interest to the top, with the exception of the "Chart Key" column that has no numeric values. The "Chart Key" column is always sorted with better designs appearing higher with respect to the design goal (i.e., maximization problems have highest values at the top, and minimization problems have the lowest values at the top; designs that violate constraints are always at the bottom). In the case of two designs having the same value for a sorting metric, a secondary sort is done with respect to the goal output value.

The plotted goal outputs in Fig. 2 each have one chart per input parameter designated as x-axes and a common output parameter for y-axes; the y-axis output parameter is different for the two sets of plots. There are two sets of charts only if there is a secondary output being considered for a task, otherwise only the goal plots are displayed. For a single design evaluation, there is one y-value corresponding to four different x-values that appear once on each set of four (or eight) charts. At the top of the plots is a reminder of the goal for a given design task. The charts also have red or gray areas depending on the constraints for a task—red areas represent the infeasible design space. When plotted, outputs for feasible designs appear as circles, while infeasible designs appear as crosses. A complete design scene consists of the interface modules as rendered in Fig. 2 passing information among one another, as well as a timer that limits the duration a user has to evaluate all of their designs before the GUI automatically moves to the next scene.

3.3 Pilot cognitive study

A pilot study was conducted to assess the usefulness of the design tool, how users generally approach the GUI tasks, and for developing data analysis methods. In the pilot study, users solved tasks with immediate or delayed feedback from the GUI (in delayed feedback condition, users received no performance output until their eighth evaluation). This approach is similar to a previous approach that investigated how delays in outputting simulation data affect human search performance (Simpson et al. 2007a). In the pilot study, users were provided a design tutorial and then solved six design tasks that had three different difficulty levels. Difficulty was based on how many outputs and constraints were present in the task statement, which correlated with the proportion of high-performance designs near the global optimum. Feedback was manipulated in the pilot study through a within-subject variation of conditions, where each user solved a task of a given type with real-time feedback and then delayed feedback. Influences of feedback were determined by aggregating all user scores for a given feedback condition and comparing them across problem types. Details on these tasks and how they were generated to balance difficulty are available in section "Design task generation and difficulty" in Appendix.

Pilot study results demonstrated that when users received immediate feedback with our myosin design tool, they found higher-quality designs than when their feedback was delayed, which suggests that the output provided by the tool is helpful during users' search process. Real-time feedback provided the greatest benefit on the most difficult task (i.e., the difference in scores among users solving with real-time versus delayed feedback was greatest on the most difficult task) and suggests that feedback from previous evaluations is increasingly more important as problem complexity grows.

Information provided by the GUI also influenced user search approaches; users with real-time feedback searched more closely to their previous best design, which correlated with more global design improvements, and supports the cognitive basis for the Compare to Best search approach in Table 1. The experimental design of the pilot study counterbalanced tasks such that half of the participants began with the easiest tasks and half began with the hardest tasks—participants did not improve on later problems. This result is important in designing future cognitive studies because it suggests that task ordering and learning across tasks is negligible, as long as users are presented diverse tasks similar to those of the pilot study. Additionally, it demonstrates that adequate training was provided during the software tutorial prior to the first task, since there was not a significant difference in user performance for a particular task depending on when they solved it during their problem set.

More details on the results of the pilot cognitive study are available in section "Pilot cognitive study with real-time and delayed feedback" in Appendix, which also demonstrates that human designers found higher-quality designs on tasks that were expected to be easier based on the number of outputs and constraints included in the task statement. However, these definitions of problem difficulties do not take into consideration that varied strategies may have more or less success depending on the nature of the space (e.g., there are many local optima, the number of feasible designs is small). Therefore, it is important to investigate the influence of strategies on successful traversal of a search space, which is considered throughout the synergistic human-agent methodology. Through creating and testing the myosin GUI, the pilot study results suggest that user search approaches can be collected and assessed that are indicative of user search approaches, thus providing a basis for collecting data in the first phase of our human-agent method.

4 Baseline cognitive study: identifying successful search trends

A baseline cognitive study was conducted to initiate our synergistic human-agent methodology by collecting data from many participants solving tasks with real-time feedback, and determining whether these trends correspond to the proposed cognitive-based search approaches in Table 1. This baseline cognitive study also provides an assessment of how users perform when not provided a formal strategy, which can serve as a basis of comparison once users are provided agent-refined strategies in the final phase of the methodology.

4.1 Baseline cognitive study protocol

Thirty-one participants were recruited to solve three design tasks of varied difficulty, and the goal of the baseline study was to have a large number of users solving each task, so that successful trends could be identified and compared with proposed Table 1 cognitive-based search approaches. The participants were all mechanical engineering undergraduates in a senior design course at Carnegie Mellon University and were compensated with course credit. The myosin design tasks given to each user are shown in Table 2 and were tuned from the pilot cognitive study (section "Pilot cognitive study with real-time and delayed feedback" in Appendix) to ensure that an initial guess of all inputs set to their median value resulted in a failed design (this is because users selected a median initial guess often,

Table 2 Three NLP design tasks for users to solve in the baseline cognitive study

Design tasks for baseline cognitive study				
Design goal	Goal constraint	Secondary constraint	Valid designs (%)	Task type
Highest avg # of attached myos	$\leq$5.7 (N_{att})	None	~50	1-Output-1-constraint
Highest filament velocity	None	System energy use $\leq$2.3 (ATP/ms)	~50	2-Outputs-1-constraint
Lowest avg # of attached myos	$\leq$5.4 (N_{att})	Filament velocity $\geq$3.2 (μm/s)	~25	2-Outputs-2-constraints

and depending on the task it could result in a high-performing feasible solution immediately). Tasks were tuned in difficulty according to the section "Design task generation and difficulty" in Appendix protocol, which requires considerations of the number of constraints on a design space in addition to the number of valid designs that meet constraints.

In Table 2, the easiest task has one goal output (representing its objective function in the myosin domain) and one constraint on the same output (limiting the maximum output, a design can have while remaining feasible). Because the task has one goal output, and a constraint on that goal output, it is referred to as a "1-output-1-constraint" task, and task types with more outputs and constraints follow a similar nomenclature pattern. Each task was limited to 4 min maximum search time, with users having to input and evaluate ten designs within that limit. All participants were able to complete all ten design evaluations within the time limit.

For the purposes of this study, users were told within a software tutorial that they were designing myosin technologies. But in the actual design tasks, the myosin information was replaced with generic variable names (e.g., "Input A") in order to focus on studying the user strategies without the influence of domain information (video of modified GUI available at http://youtu.be/-N–Jopb4DA). Outputs were also renamed with generic variable names (e.g., "output B"). The pairing of each myosin design variable and each generic name was changed for each task, to minimize user familiarity with how each input affected outputs. More generally, in this cognitive study, we are investigating how humans navigate "generic" complex spaces with black box simulators and how to best provide strategic guidance as they search the nonlinear and highly coupled input–output space; therefore forming an approach that is tied to the myosin domain, yet the strategies developed could apply broadly to many different nonlinear complex system domains.

4.2 Tabulating user search behaviors

All 31 users solved the design tasks under the same real-time feedback conditions, and information about users' designs and evaluations was recorded throughout each user's entire search process. The post-processing of recorded data to find user search trends was informed by the pilot study and the proposed cognitive-based approaches in Table 1, such as determining how often users utilized univariate searches. All information and subsequent calculations to interpret a user's search process are presented in Table 3, which will be used as a reference throughout this section and reflects the recorded data of one user solving the 2-outputs-2-constraints task. The post-processing calculations in Table 3 were unknown to the user; users were only presented information via the GUI as noted in Sect. 3.1.

When a design was evaluated, its objective function value was displayed to the user as its calculated value corresponding to the myosin domain, but in our table we assessed designs via their relative objective functions compared with the global optimum for the task. The global optimum for a problem was always 1 and all designs worse than the global optimum were given a score between 0 and 1 as a proportion of the design's objective function and that of the global optimum (the reciprocal was used for minimization tasks). For instance, if a user had a design evaluated at 2 μm/s, and the global maximum was 4 μm/s, then the user would be presented 2 μm/s and the calculated relative objective function for Table 3 is 0.5 (no units). If an evaluation violated a constraint, it was given a relative objective function score of zero. By scoring all tasks according to these scales, comparisons could be made across tasks based on relative scores, such that a higher score always indicates a better design regardless of whether the task requires a minimization or maximization.

Although all failed designs were given relative objective functions of zero, and the GUI informed users that all failed designs were equally bad, our post-processing calculations require identifying a user's best design among all failed designs if the user has not yet input a valid design. This was calculated through determining how badly a design violated constraints and assessed for each constraint separately. For instance, if a user's evaluated design had a filament velocity of 5.5 μm/s, and the secondary constraint for the task stated that filament velocity had to be <4.0 μm/s, then the secondary constraint violation would be given a value of 1.5 in the table. The magnitude of a constraint violation was always calculated as an absolute value, such

Table 3 Search data recorded from one user and subsequent post-processing calculations

Design #	Relative objective function	Goal constraint violation	Secondary constraint violation	Design input variables				Previous best design input variables				Input Distance from Best	No. of variables different from best	Has knowledge of design variable				Applied knowledge greedily?
				Lever	k_{on}	k_{off}	N_{myo}	Lever	k_{on}	k_{off}	N_{myo}			Lever	k_{on}	k_{off}	N_{myo}	
1	0	1.82	0	3	3	3	3	–	–	–	–	–	–	–	–	–	–	–
2	0	0	1.35	2	5	2	1	3	3	3	3	6	4	×	×	×	×	–
3	0	11.62	2.11	2	3	1	5	3	3	3	3	5	3	×	×	×	×	0
4	0	0.57	0	4	2	4	4	3	3	3	3	4	4	×	×	×	×	0
5	0	0.87	0	5	2	4	4	4	2	4	4	1	1	YES	×	×	×	0
6	0.41	0	0	4	1	4	4	4	2	4	4	1	1	YES	YES	×	×	0
7	0.48	0	0	4	1	5	4	4	1	4	4	1	1	YES	YES	YES	×	0
8	0.58	0	0	4	1	5	3	4	1	5	4	1	1	YES	YES	YES	YES	0
9	0	0	0.81	4	1	5	1	4	1	5	3	1	1	YES	YES	YES	YES	1
10	0.72	0	0	4	1	5	2	4	1	5	3	1	1	YES	YES	YES	YES	2

Higher relative objective functions indicate better designs

that a value closer to zero is always closer to satisfying a constraint, regardless of it being an upper or lower bound. Users could have utilized such information during their design process through assessing how badly failed designs violated constraints through viewing the charts or tabulated output.

A user's best design was then determined as the one with the highest relative objective function, or if no designs were within constraints, the best design was the one that violated the secondary constraint the least. If the secondary constraint was not violated, or did not exist, the constraint on the goal output was used as a basis of comparison. The secondary constraint was considered more important in limiting the best design because all tasks were tuned such that the secondary constraint was actively constraining the global optimum (section "Design task generation and difficulty" in Appendix). The user's best design was identified during each point in the search process, therefore enabling pairwise comparisons of a user's current evaluation against their previous best design for each point in the search process.

Because each evaluated design was input via four myosin variables with five possible values (details noted in Sect. 3.2), each of these input variables were given a value of 1–5 in the table. The difference among inputs from a user's current evaluation and previous best design was calculated through determining the absolute value of the difference between each variable among the two sets of inputs, and then summing all of the differences. Finding this distance can indicate whether a user was searching near their previous best design and reflects the Compare to Best approach in Table 1. Determining the number of variables different from the best design configuration required pairwise comparisons of each myosin input for the current evaluation against that of the previous best design and summing the total, therefore leading to a value of 4 or lower.

The next set of post-processed data was based on cognitive considerations of users learning and applying knowledge within a search process—higher or lower values of these variables are not necessarily better or worse, but serve as metrics for comparing the best and worst design searches in order to find patterns in the approaches utilized by the best human designers. We considered these calculations in the context of univariate approaches, because they are effective in knowledge inference/scientific reasoning contexts (Chen and Klahr 1999; Kuhn et al. 2008), as suggested in Table 1 by the univariate search approach. A variable was counted as learned if it had been used in a univariate fashion from any other past evaluation. Once a user learned a variable, the knowledge was considered to be retained throughout the rest of the search.

It was then calculated whether the user implemented learned knowledge usefully in the rest of the design process

according to greedy search logic, which could potentially mean a user is applying knowledge from their short-term memory as indicated in Table 1 by the Learn and Apply search approach. A useful univariate search was considered an evaluation if the user implemented a univariate search from the previous best design using a learned variable and altered the variable in a direction that would improve its objective function or overshoot a constraint in an attempt to improve the objective function. If the previous best design was in violation of constraints, users were considered to have utilized knowledge properly if they moved a failed design toward being valid by using a univariate search that lessened the constraint violation (e.g., if a design violated a constraint because its filament velocity was too high, and decreasing lever arm was known to decrease filament velocity, a correct knowledge application was counted if the user decreased the myosin lever arm length using a univariate search). In cases of a failed design for the 2-outputs-2-constraints tasks, assessment of proper knowledge application was done with respect to the secondary constraint initially because it is considered the more important constraint to avoid, as it limits the global optimum. Knowledge application with respect to avoiding the goal constraint was only considered if the best design was infeasible but had no secondary constraint violations. Through tracking these relationships for each user, a basis of comparison among user search behaviors can be conducted that provides cognitive insights for how humans effectively search complex design spaces.

4.3 Identifying successful user search trends

The tabulation of user searches is important, because it allows for comparing user searches and aids in determining trends that most often lead to high-quality final designs. For each task, data was analyzed by separating user results into three categories based on the best relative objective function found during their search; the top 25 % of searches were placed into the Best search category, the worst 25 % of searches were placed into the Worst search category, and all other searches were placed in the Med (median) search category. Because each task was assessed independently of others, the users that produced the best searches for one task did not necessarily produce the best searches for other tasks. The comparisons of these three categories for each task and other examined trends are presented in Fig. 3.

The results in Fig. 3a shows that the searches in the Best grouping were always of greater quality than a random solver. The random solver was programmed to select designs sequentially under the same constraints as humans (i.e., only ten design evaluations allowed, all designs must be unique) by randomly choosing values for each design variable for each new evaluation. Searches in the Med grouping were slightly better than a random solver for the 1-output-1-constraint task and comparable with the random solver for the more difficult tasks. The searches in the worst grouping always performed worse than the random solver. In considering all tasks, the searches from the best grouping always averaged to a relative objective function of about 0.9 or greater, and the difference between searches in the Best and Worst grouping grew as task difficulty increased. Thus, the gap between the best and the worst searches grew with task difficulty, which is similar to the pilot cognitive study (section "Pilot cognitive study with real-time and delayed feedback" in Appendix).

The first considered post-processing trend was how far users searched in comparison with their previous best design, which corresponds to the Compare to Best cognitive approach in Table 1. This was calculated for each task, by summing the Input Distance from Best value from Table 3 across all of a user's evaluations and then averaging them for all users in a category (same Best, Med, and Worst categories as in Fig. 3a) and is plotted in Fig. 3b. The results show that in all tasks, the searches in the Best grouping had lower average differences than searches in the Worst grouping. The greatest difference between the Best and Worst searches occurred on the most difficult design task (2-outputs-2-constraints). The user trends were also compared with the distances a random solver would produce during its search. For the 2-outputs-2-constraints task, searches in the Worst grouping varied much more widely from their previous best design than would be expected by a random search, suggesting they may not have had a methodical approach to searching the space or that their approach was highly ineffective. When referring back to Fig. 3a, the searches in the Worst grouping also had many scores near zero in their 2-outputs-2-constraints task, thus suggesting that when user designs were infeasible (thus having a score of zero), users were more likely to widely search the design space. These trends suggest that better searches are a result of users choosing new inputs that are similar to their previous best input, rather than widely searching the design space.

Next, the total number of times a univariate search was used in relation to a previous best design was summed for each user and averaged across all users for each category in Fig. 3c, which is indicative of the univariate search approach in Table 1. The categories of Best, Med, and Worst still reflect the same groupings as originally created for Fig. 3a. Results in Fig. 3c shows that for the least and most difficult tasks, the univariate search in relation to the best design was used much more often in successful searches than the worst searches. For the 2-outputs-1-constraint problem, there was not a significant difference among the three search categories, suggesting that univariate searches

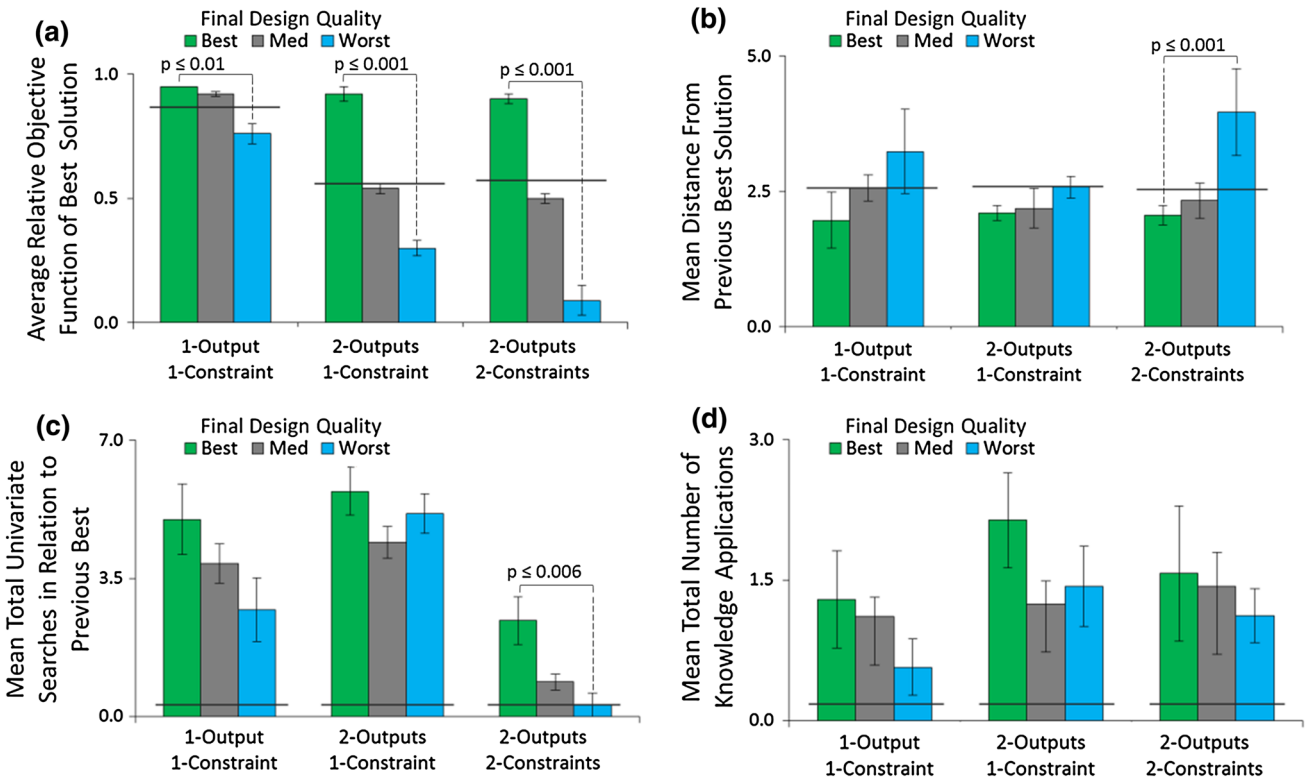

Fig. 3 Performance and trends of users separated into Best, Med, and Worst search categories for each design task. The output trends considered were **a** the mean best relative objective function, **b** the mean difference between a user's design inputs and previous best solution, averaged over all evaluations, **c** the mean number of times users implemented a univariate search strategy using their previous best design as a point of reference for a given search, and **d** the mean total times users applied knowledge. All *error bars* represent standard error of the mean; significant ($p < 0.05$) differences are indicated by reported *p*-values on charts for comparisons among the Best and Worst user measurements. *Horizontal lines* indicate random solver results

might not be beneficial for that particular task. In all cases, user implementation of the strategy was much higher than was achieved by the random solver and suggests that univariate searches are a deliberate cognitive-based strategy, rather than a pattern that emerges by chance.

The final set of analyses examined whether users had applied knowledge according to a greedy search logic once they had learned (or at least had the opportunity to learn based on evidence from their search process) how an input parameter affected the output through isolating it in a univariate search in relation to any previous design. This measurement is indicative of users searching with the Learn and Apply approach in Table 1. The number of times knowledge was applied according to greedy search logic rules (Sect. 4.2) was then counted and averaged for all users within a category. This data is plotted in Fig. 3d for each task type, using the same categories for Best, Med, and Worst.

Although Fig. 3d results show that all users appear to have applied knowledge much more often than would be expected through a random search, the results show a high degree of variability across all groups. Therefore, these observed group differences are only suggestive trends. However, in all tasks, results show the same trends of knowledge implementation being positively correlated with more successful searching. This, as well as the other two trends identified in Fig. 3b (to search near a user's previous best designs) and Fig. 3c (to use univariate searches in comparison with a user's previous best designs) suggest that better searches are the result of deliberate decisions in the design process. Such findings support the notion that cognitive-based search approaches may be highly effective in traversing complex system spaces.

Figure 3 findings, when considered with Table 1 cognitive-based search approaches, can be used to develop formal search approaches that agents can test at a much faster rate than could be achieved through further human studies. Three search strategies of Stochastic From Best, Univariate From Best, and Univariate Learn and Apply were developed through consideration of the data and are presented in Table 4.

The strategies presented in Table 4 are specifically search strategies, as opposed to other strategies associated with the design process (e.g., ideation, testing.) and therefore have specific rules. The first strategy, Stochastic From Best, was inspired by the best designers in Fig. 3b who often selected designs that were near their previous best design. In this

Table 4 Agent strategies informed from user search successes

Strategy name	Strategy rules
Stochastic From Best	One or more design variables in comparison with the best known solution are slightly perturbed
Univariate From Best	One design variable in comparison with the best known solution is perturbed
Univariate Learn and Apply	One design variable in comparison with the best known solution is perturbed. Subsequent perturbations are based on parameter correlations

strategy, a user or agent must choose their next design through changing any number of variables as long as it is within the maximum difference allowable between designs. The Univariate From Best strategy was inspired by the best designers in Fig. 3c, who often utilized univariate searches. In this strategy, one design variable must change in relation to the best known design for each new evaluation. The final strategy was inspired by the trends observed in Fig. 3d in which better searches were related to learning and application of parameter relationships. In this strategy, a user or agent must utilize univariate searches to learn a parametric relationship among one input and the output(s) for a problem and then use univariate searches to change that parameter in an effort to improve their design using their acquired knowledge. These rules have many variations that can be quickly investigated with agents, which is the next major step of our human-agent methodology.

5 Agent study: refinement of human strategies

After identifying trends in how users with the highest search scores traversed the design space, software agents were programmed to automate task solving by using Table 4 strategies inspired by the best user searches. Computational software agents were constrained to the same rules as human designers so that the findings could be directly compared with user solutions, and successful agent strategies could be returned and implemented by users.

5.1 Finding effective strategies with software agents

Each software agent was a computational object that has access to all the same information available to human designers via the tabulated results table in the GUI (e.g., the values of previous design inputs and performance outputs). Agents searched the design space by assessing the available information and then choosing a next input that followed a set of rules based on that agent's preferred Table 4 strategy. Because there was no initial information available for an agents' first guess, each agent was given a preferred choice in initial design (e.g., random inputs, or choosing median design inputs). In subsequent steps, agents always followed the rules of their preferred strategy, unless there was no combination of design inputs that would enable them to adhere to their strategy—in these cases, the agent input a random design that had not yet been evaluated. Because the agents are utilizing many stochastic search rules for finding designs (such as choosing which design input to vary randomly), and good strategies can lead to poor results based on chance (or the reverse), each agent repeatedly solved a task 2500 times, and results were aggregated to generate an average result with negligible error.

The findings from the baseline cognitive study led to three different cognitive-based search strategies, which are presented again here in the context of how agents deploy each strategy. The rules for the Stochastic From Best strategy are that a new design cannot have a difference larger than a given amount (based on the agent's preference) in comparison with the previous best design, but any number of variables are allowed to change (i.e., any number of variables could change, but the total distance across variables is constrained using the same distance criteria as in Table 3 post-processing calculations). For the Univariate From Best strategy, agents randomly chose a design input to change in comparison with their previous best design.

The Univariate Learn and Apply strategy utilized a stochastic univariate search strategy for each of the four design inputs initially, with design variables chosen in random order until a univariate search had been conducted once with respect to each input variable. When a variable had been chosen for a univariate search, an agent had an equal chance of randomly choosing any other value for the variable other than its current value (that did not duplicate a previously evaluated design). The agent then applied knowledge via a greedy local search that followed the same rules of successful knowledge implementation by users in Sect. 4.2. To restate those rules in the context of agents, if a design was feasible, the agents had to make design changes that would result in a better objective function value or overshoot it. If the design was failed, the agent had to make design changes that would result in a lower constraint violation.

The mean best relative objective functions found by agents utilizing these three strategies are presented in Fig. 4 for each of the three Table 2 tasks and are compared with a random solver. For all agent strategies, the initial guess was chosen randomly and the Stochastic From Best strategy reflects a preferred distance of two from the previous best design, which is consistent with the best approach found when extensively testing all agent strategies and their variations in section "Pilot cognitive study with real-time and delayed feedback" in Appendix.

For the 1-output-1-constraint task, all strategies performed better than a random solver, with the Univariate

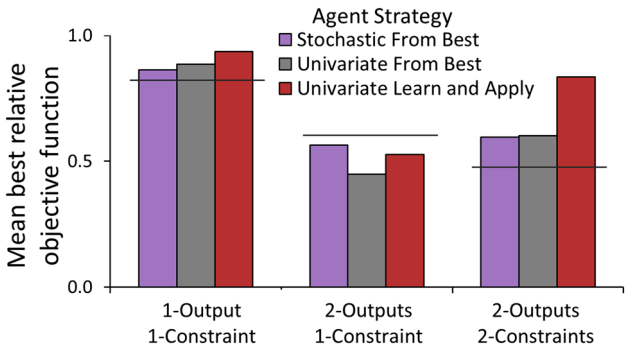

Fig. 4 Mean best relative objective function found by agents for each task. Error for all mean agent searches is negligible. *Horizontal lines* indicate random solver results

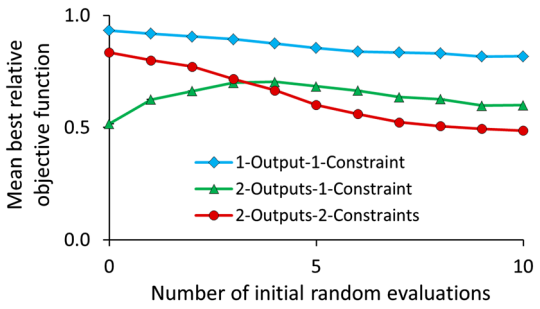

Fig. 5 Mean best relative objective function found by agents utilizing the Univariate Learn and Apply strategy as number of initial random evaluations is varied for each task. Error for all mean agent searches is negligible

Learn and Apply strategy outperforming the other two strategies. For the 2-outputs-1-constraint task, the random solver outperformed each of the strategies, which is consistent with this task having the weakest trends in differentiating the best and worst searches by users in Sect. 4.2. In the 2-outputs-2-constraints task, the Univariate Learn and Apply strategy produced higher-quality solutions than all other strategies, although the other two strategies did perform better than a random search. These results suggest that the Univariate Learn and Apply strategy has the most consistent performance, although another approach could improve search performance in the 2-outputs-1-constraint task. Because random evaluations performed well in this space, a refined strategy could utilize a combination of random evaluations early in the design search followed by utilization of a directed strategy, which is similar to past successes in combining stochastic and deterministic optimization approaches (Aladahalli et al. 2007). Such a strategy combines the exploratory nature of stochastic searches early in the design process, followed by a directed convergence later in the design process, thus forming a broad search approach. This broad approach may specifically perform well on the 2-outputs-1-constraint tasks because it has both a large feasible design space (50 %) and an unbounded objective function that is constrained by a secondary output, thus producing a design space that would have many local optima that perform much worse than the global optimum. Because there are many local optima, a broad exploration is beneficial in the search process because it facilitates the finding of strong candidate designs before applying knowledge for convergence on a final design.

5.2 Assessment of broad early search

In order to improve the search effectiveness in the 2-outputs-1-constraint task, it was proposed that a broad search early in the design process would provide exploration that would facilitate the finding of the global optimum design. To assess the most beneficial number of stochastic selections, agents were programmed to utilize the Learn and Apply Strategy for each Table 2 task and have a number of initial random guesses ranging from zero (i.e., the same Learn and Apply Strategy from previous section) to ten (effectively a random solver) and results are presented in Fig. 5. After their initial random guesses, all agents utilized the Learn and Apply strategy for their remaining evaluations.

As the number of initial random evaluations increases, the best found solution declines for the 1-output-1-constraint and 2-outputs-2-constraints tasks, but increases initially for 2-outputs-1-constraint task before decreasing. Initial random searches may not improve the 1-output-1-constraint task because the objective function itself is constrained, thus making it a bound problem, and therefore, all local optima will be very close to the global optimum and any approach that quickly converges on the constraint will always result in a good solution. The 2-outputs-1-constraint task may have benefitted from initial random searches in comparison with the 2-outputs-2-constraints task, because the 2-outputs-1-constraint task has twice as many designs that do not violate any constraints (because each constraint halves the design space given how these tasks were created, see section "Design task generation and difficulty" in Appendix). This means there are many more local optima that could cause earlier convergence on a sub-par solution in the 2-variables-1-constraint task. By contrast, the 2-outputs-2-constraints task has more chances of random designs resulting in infeasible solutions, which helps direct converging algorithms toward a high-performing feasible solution. Because there are so few valid designs for the 2-outputs-2-constraints task, as long as an agent finds a valid design that converges on the active constraint, it is likely to be higher scoring than a random design. These results suggest that optimal strategies vary by task type, and that a robust strategy for many types of problems could be investigated using the agents, which is presented in the next section.

5.3 Expert knowledge application and robust design strategies

To facilitate identification of a robust design strategy, three more tasks were generated (Table 5). More tasks are required to fully assess a strategy because one strategy could by chance work well on a given problem type based on initial conditions chosen, or other random chances, which obscures generalizability. To assess whether a strategy is robust, its performance is aggregated using all six tasks across Tables 2 and 5. Tasks in Table 5 were tuned in difficulty according to the same rules as Table 2 tasks.

Because these strategies will be eventually returned to users and users may have varying levels of a priori knowledge, strategies also require assessment in terms of how differing levels of domain knowledge affect search quality. For instance, an expert designer may already know all parameter relationships and could therefore bypass the learning stages in the Univariate Learn and Apply strategy and immediately begin applying their knowledge in a greedy local search. However, this may further direct the search toward early convergence, which could impede or improve performance depending on the task. Therefore, it is important to investigate each approach with agents to determine how a priori and learned domain knowledge affects performance on each task.

To investigate how the number of known parameter relationships (referred to as increments of domain knowledge) influence search quality, the Univariate Learn and Apply strategy was conducted with different amounts of maximum domain knowledge that could be known by an agent. Then a new agent strategy, termed Univariate Expert Application was developed, which reflects agents knowing whether a design variable has a positive, negative, or no correlation with each output parameter before they begin a task, such that they can immediately apply knowledge after their initial solution without an initial learning phase. This strategy simulates a user with expertise in the myosin domain. Agents solved these tasks with different amounts of maximum domain knowledge, and Fig. 6a shows the mean best relative objective function found for each strategy for both the expert and learning approaches when results are aggregated across all Tables 2 and 5 tasks.

For both expert and learning strategies, having higher domain knowledge improves design task performance, and there is a negligible difference between expert and learning approaches on final design quality (Fig. 6a). A second perspective of the data was considered for the highest performing cases of each strategy (where maximum domain knowledge is four) in Fig. 6b, by plotting the mean best relative objective function after each evaluation during the search process. Results show expert domain knowledge provides a much faster convergence (Fig. 6b). The results also demonstrate a growth after all knowledge has been gained by the Univariate Learn and Apply strategy, when knowledge is being applied to search the space much more effectively than a random solver or its initial stochastic learning period.

A larger set of strategies is explored in section "Summary of best agent strategies" in Appendix, where agents were configured with many different preferences in their search approaches, thus resulting in the assessment of over a thousand strategic variations in searching these spaces. Results from considering the large pool of agent searches suggested that the Univariate Learn and Apply and Univariate Expert Application strategies were superior to other approaches. The best Univariate Learn and Apply strategy, when aggregated across all tasks, had agents configured with an initial starting guess of all design inputs set to their median and no random evaluations. Including a random evaluation after the initial evaluation only reduced the overall score by $\sim$0.01, while improving the 2-outputs-1-constraint problem in Table 2 by $\sim$0.04. The best Univariate Expert Application had its most successful configuration when its starting guess had all design inputs set to median and no initial random evaluations. When three random evaluations were included after the initial guess, it only lowered the aggregated score across all tasks by $\sim$0.02, yet increased the 2-outputs-1-constraint task by $\sim$0.12.

Thus, for an unknown set of design tasks, the most robust approach may include random evaluations if there were many problems similar to the 2-outputs-1-constraint type task. If there were many 1-output-1-constraint type tasks or 2-outputs-2-constraints type tasks, then the most robust approach would have no initial random evaluations.

Table 5 Set of three additional NLP design tasks for testing robustness of agent strategies for additional, yet similar, design tasks found in Table 2

Design tasks for robust and expert strategy assessment				
Design goal	Goal constraint	Secondary constraint	Valid designs (%)	Task type
Highest system energy use	$\leq$2.3 (ATP/ms)	None	$\sim$50	1-Output-1-constraint
Highest filament velocity	None	Avg # attached myos $\leq$5.8 (N_{att})	$\sim$50	2-Outputs-1-constraint
Lowest avg # of attached myos	$\leq$5.4 (N_{att})	System energy $\leq$1.2 (ATP/ms)	$\sim$25	2-Outputs-2-constraints

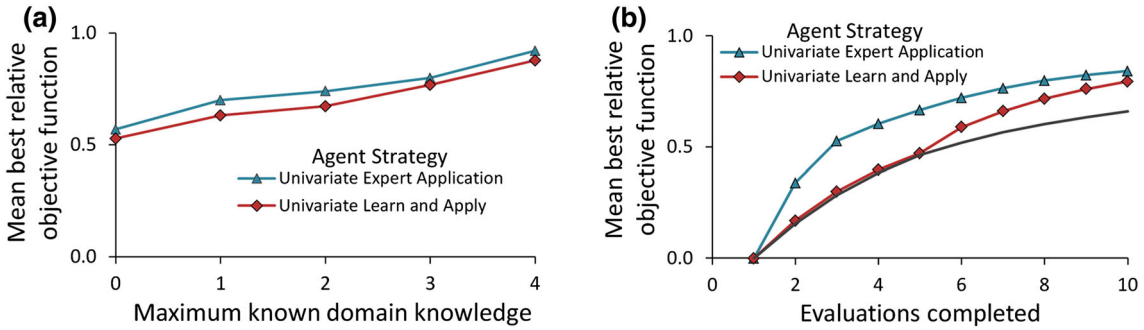

Fig. 6 Contrasts among Univariate Learn and Apply and Univariate Expert Application approaches. The mean best relative objective function found by a search (**a**) as a function of maximum knowable domain knowledge and **b** as each evaluation is completed for a comparison of the Univariate Expert Application and Univariate Learn and Apply strategies when maximum domain knowledge is four. Error for all mean agent searches is negligible. *Both lines* in part (**a**) are nearly equivalent and the dark line in part (**b**) indicates result from a random solver

These results demonstrate that through iterations with agents, many successful cognitive-based search approaches can be found for effectively solving these tasks, with many of the strategies extending beyond what can be proposed prior to user studies (such as including random evaluations after an initial guess, which was not included in Table 1 as a potential approach, but rather discovered as useful when investigated with the agents). Our next step was to take a portion of these highly successful strategies and provide them to users and then determine whether user implementation of the agent-derived strategy improved over users who approached the tasks with no formalized strategy provided.

6 Final cognitive study: users aided by agent-derived strategies

We conducted a final cognitive study, with a goal of determining whether users would benefit from using the most effective strategies found by the agents. This cognitive study consisted of a control problem, followed by users implementing the Univariate Apply and Learn strategy and Univariate Expert Application strategy. Depending on their study condition, users searched the space with or without early random evaluations. The GUI was modified to facilitate user implementation of these strategies and our expectation was that users would perform similar to the agents, and that the strategy-supported user performance should exceed the performance of users from the baseline cognitive study, who were not provided a formalized strategy.

6.1 Experimental protocol

The participant population for the third cognitive study consisted of 30 senior undergraduate and graduate students in engineering and science-based disciplines enrolled in a Cellular Biomechanics class at Carnegie Mellon University. The performance of users in this final study was compared back with the users from the baseline cognitive study in order to evaluate the benefit of the agent-refined strategies. Although this Cellular Biomechanics class had a mix of students from different academic levels and disciplines in comparison with the baseline study (all senior mechanical engineering students), the overall design performance of subgroups was very similar, and the strongest performing subgroup was the senior mechanical engineering students. Therefore, this new class serves as a good conservative comparison group.

Participants were compensated with course credit and randomly placed into one of two study conditions. Before solving the design tasks, all users went through a self-guided software tutorial. Each task was also limited to 4 min maximum search time, with users having to input and evaluate ten designs within that limit. All participants were able to complete all ten design evaluations within the time limit. In both conditions, users solved the same three tasks from the baseline cognitive study (Table 2) and then the tasks from Table 5. For all tasks, the GUI forced users to input their first evaluation as the median of all design input values. An initial median guess for each participant was chosen for the cognitive study to enforce consistency among users, and because it was consistent with $\sim$65 % of user searches beginning with a median guess during the baseline cognitive study.

In the first condition, users implemented Direct variations of strategies: Direct users solved the same tasks as in the baseline study (shown in Table 2) using the Univariate Learn and Apply strategy, and then the additional tasks explored in the final study (shown in Table 5) using the Univariate Expert Application strategy. Given the lack of task order effects in the pilot study, the sequencing of tasks should not matter, but we conservatively placed the same tasks from the baseline cognitive study first to make a conservative comparison with that study's results possible.

In the second condition, users implemented Broad variations of the two strategies; Broad users also solved the first three tasks using the Univariate Learn and Apply strategy, and then Table 5 tasks using the Expert Application strategy, but the GUI generated three random solutions for their second, third, and fourth evaluations. (Note, to reduce the chance of randomly generated solutions that are univariate searches, each randomly generated solution was required to have at least two input values different from all previous inputs). Three random inputs following the initial input for the Broad strategy were chosen as a strategy because it was consistent with high-performing strategies determined by the agents (section "Summary of best agent strategies" in Appendix).

For Table 2 tasks, users solved the problems with the generic variable names and sliders (as implemented in the baseline cognitive study in Sect. 4). Therefore, any domain knowledge differences among participants should be negated as they were presented an interface that had minimal contextual information. In contrast to these first three tasks, where inputs/outputs were labeled generically, users solved Table 5 tasks with myosin information and graphics presented by the GUI as depicted in Fig. 2—all graph axes were labeled with myosin input/output information, the myosin/system renderings were presented to users, and the results table had myosin labels. This addition was included because users solved these three tasks using the Expert Application strategy, which is based on users being given relevant domain knowledge prior to the task and applying that knowledge in a manner that is representative of expertise in the domain. The assumption that these users can operate very close to experts in the domain is supported by recent GUI studies that demonstrated novices with minimal instruction prior to GUI tasks can perform similar to experts (Wolf et al. 2011).

Several modifications to the GUI presented in Fig. 2 were implemented to support this final cognitive study (video available at http://www.youtube.com/watch?v=3psPSKYl9WE). A table was added that summarized known relationships a user uncovered during their search process, or that summarized a priori parameter relationships in the case of the Expert Application strategy. The table summarized how increases in a known input affect the goal output and secondary output (if a secondary output was present in the task statement), which reduces the cognitive load by having less demand on a user's ability to remember all relationships. For the Univariate Learning and Application strategy tasks, the knowledge table initially stated that all input–output relationships are unknown, but filled in the corresponding relationships once the corresponding univariate searches were conducted for design variable.

The GUI was also modified to facilitate user searches in reference to their best design through adding highlights under each slider to indicate a user's previous best design. The ticks under each slider were redrawn after each evaluation to inform the user of which potential designs are valid for their next evaluation; only variable values that were consistent with valid designs for the given strategy had ticks drawn underneath them. The GUI did not allow a user to input a design that violates the necessary strategy and, if a user attempted to input such a design, the GUI provides contextual information for why a user's attempted design does not meet the rules of the strategy. These features of the GUI reduce the amount of cognitive load on the user, thus enabling them to concentrate more on making important design decisions instead of trying to remember knowledge, their best design configuration, or the rules of the strategy. These changes also reduce possible user interface frustrations. These modifications also help control the cognitive study by enforcing users to implement the strategies required for a given task based on their study condition.

6.2 Direct and broad strategies cognitive study results

Prior to separating groups into different conditions, a control task was given to determine whether users in each population had nearly equal proficiency in solving a task. The control task was a 2-outputs-1-constraint task, with an objective function of highest system energy use and a secondary constraint of filament velocity not exceeding 3.3 μm/s. The results showed that one of the two populations of users outperformed the other slightly when comparing the best design found as an average for each group. Therefore, the best users were removed from the analysis in the high-performing group until it reached equal performance to the least performing group. Equality occurred once the three highest performing users were removed; all subsequent results from this cognitive study are considered with respect to having these users removed from the study analysis.

Each population was then randomly assigned the Broad or Direct strategy and trained to utilize their respective search strategies in the context of the modified GUI via the software tutorial. The users first solved Table 2 tasks and the mean relative objective function of the best solution by the end of each search was averaged for each population of users. These are plotted for each task in Fig. 7a, in addition to showing the performance of agents utilizing the same strategies, a random solver as a basis of comparison, and the user results from the baseline cognitive study (labeled as "Human None" to reflect they were provided no formal search strategy).

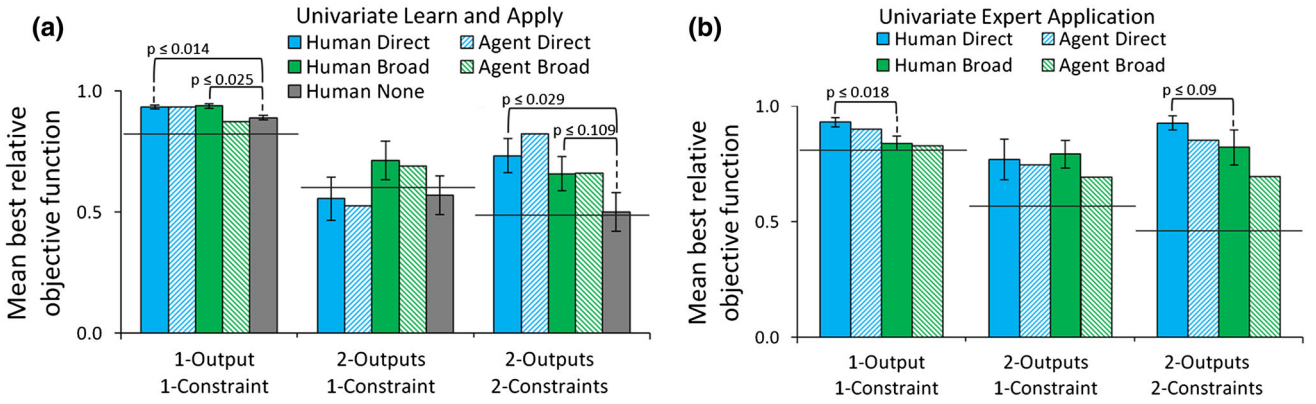

Fig. 7 Mean best relative objective function of agents and users for **a** Table 2 design tasks when using the Univariate Learn and Apply strategy with Direct and Broad approaches and **b** Table 5 design tasks when using the Expert Application strategy with Direct and Broad approaches. All *error bars* represent standard error of the mean; significant ($p < 0.05$) and trending differences are indicated by reported p-values on charts for comparisons of user collected data. *Horizontal lines* indicate random solver results. Error for all mean agent searches is negligible

Focusing first on the tasks shared with the baseline study (Fig. 7a), the Human Broad strategic condition outperformed the prior user performance for all three problem types. The Human Direct strategy outperformed the prior user performance for two of the three tasks. The one exception was the 2-outputs-1-constraint problem with direct search, and there the agent simulation had suggested it would not be very effective. Thus, through considering all pairwise comparisons of user performance with their respective agent-refined strategy, there is clear evidence pointing to the benefit of providing designers with strategies refined by agents, particularly ones the agent simulations suggest will be effective for the problem at hand.

When considering all six tasks (Fig. 7a, b), there was generally a small difference between the relative success of agent search and relative success of humans provided the same strategy. Of the twelve possible pairwise comparisons among humans and agents, when considering the mean user score with standard error compared with the agents, there were four tasks where differences were large. In three of the tasks (Table 2: 1-output-1-constraint, Table 5: 2-outputs-1-constraint, and Table 5: 2-outputs-2-constraints), humans greatly outperformed agents in the broad condition. The increase in performance from the broad users over the agents was possibly because users were able to assess sensitivities and effectively choose orderings of when to change a given design variable, whereas agents stochastically choose a design input and how far it changes.

In the 2-outputs-2-constraints task for the Univariate Learn and Apply strategy, there was a large difference between the users utilizing the direct strategy and their respective agents, in which the users utilizing the direct strategy underperformed (the only significant user underperformance case). These differences were likely caused by the ordering of input options in the GUI for that problem, because users manipulated the detachment rate of myosins $\sim$50 % of the time initially (rather than the 25 % occurrence via the unbiased agents). The increased likelihood of users manipulating this input occurred because all input–outputs were labeled generically (i.e., no contextual myosin information), and the slider corresponding to detachment rate was located in the upper left of the screen; generally, when given no contextual information, users select options top to bottom, left to right (Nielsen 2006; Shrestha and Lenz 2007). The feasibility of this being a poor starting location is supported when considering that agents with different initial guesses was explored (such as beginning searches with all input variables minimized or maximized) and often performed worse compared with agents that utilized random initial guesses or median guesses (section "Summary of best agent strategies" in Appendix).

Because there are differences in strategy performance for each task and approach, the most robust strategy across all task types was determined by averaging the mean relative objective function for all six tasks in Fig. 7 for the Human Broad strategy, Human Direct strategy, Agent Broad strategy, and Agent Direct strategy independently. Based on these averages, both the Human Direct and Human Broad strategies had an average score of $\sim$0.8, suggesting both approaches are equally successful with consideration to these six tasks. Agents, on the other hand, performed better with the direct approach across tasks and had a score of $\sim$0.8, while the Agent Broad strategy only scored $\sim$0.74 when averaged across all tasks. This suggests that human biases may provide an advantage in searching tasks using a broad approach when compared to agents, although comparisons must be made with caution as there is uncertainty in the human measurements. The broach approach may also be the best human strategy for complex systems as it performs well across many different

problem types and is robust to possibly very difficult and counterintuitive tasks, such as the 2-outputs-1-constraint task from Table 2.

7 Discussion

The utility of the design software and solution approaches in complex system design are discussed, followed by an examination of strategy and domain knowledge influence on search quality for varied tasks, and finally how the human-agent synergistic approach may extend to other complex system domains and engineering decision-making in general.

7.1 Utility of GUI for facilitating human-based design

The GUI was developed to remain extendible to complex systems beyond the myosin application. In the simplest case, as long as the system may be represented in NLP form, it could be translated into the design interface by changing the names of parameters that appear, the mathematical equations calculated behind the scenes, and the rendered graphics. In cases where there are more input and output parameters than the amount considered in this study, users could have selections of which inputs and outputs to view at a given time, rather than having all inputs and outputs appear on the screen simultaneously. The modularity and ease of implementing varied forms of the GUI was demonstrated throughout this study through different feedback conditions, representations with generic input–output names, and contextual help for aiding users in implementing the agent-refined strategies.

The GUI guided users in implementing strategies during the final cognitive study, which could make it a useful training tool for teaching humans effective search strategies and for reducing the effects of human biases. One of the aids from the GUI was its generation of random designs for users during the broad strategy conditions, which reduced the amount of cognitive bias a user could introduce into the broad search. The GUI also provides contextual information to aid users in implementing their knowledge-based univariate strategies. Similarly, it would be possible to further support multivariate user searches through contextual hints provided by the GUI.

The GUI in this study was restricted in many ways because it had to present a controlled environment that facilitated cognitive studies and post-processing of human results. However, future implementations could enable users to view a greater amount of information and visualizations. Further, each of the NLP design tasks used in this study represents a potential myosin technology, and this tool could enable a designer to quickly formulate a set of functional requirements and explore what potential myosin system embodiments could fulfill those requirements. Future work could explore other possibilities of extending the GUI for more practical design use, especially extending to higher-dimensional spaces.

7.2 Effectiveness of cognitive-based strategies

Of the considered cognitive-based search approaches (Table 1), the highest performing searches utilized domain knowledge either through learning or applying a priori knowledge (that simulates an expert). Knowledge application in this paper was always conducted according to greedy search logic for fast convergence, while stochastic search processes were utilized to facilitate broad early design space exploration. In the 1-output-1-constraint tasks, which were generally the easiest search spaces, any approach that leads to convergence on the constraint performs well, because the constraint is indicative of the global optimum for the problem (otherwise known as a bound problem). Therefore, greedy univariate searches will always perform well on the 1-output-1-constraint tasks. Greedy searches also worked well for the 2-outputs-2-constraints task for similar reasons, in which convergence on a constraint was likely to result in a high-quality design because a large portion of the design space resulted in infeasible designs (only 25 % of designs were valid), therefore leaving fewer local optima for the greedy algorithm to find.

In the 2-outputs-1-constraint task from Table 2, greedy searches were only effective if random evaluations were introduced early in the search. Such early exploration may have been necessary due to the nature of the 2-outputs-1-constraint design space in comparison with the other tasks; the 2-outputs-1-constraint task has a large feasible design space (50 % of designs are valid) and a constraint on the secondary output, which forms a design space that may have many pareto optimal solutions along the constraint. Therefore, fast convergence on an optimum leads to a trade-off among the goal and secondary output such that changing one design variable at a time will always result in a worse solution, even though the global optimum is not currently found. Therefore, the introduction of a broad early search aids in finding good candidate solutions across the entire design space before fast convergence on a local optimum by the Univariate Learn and Apply Strategy (in some cases, the local optimum may also be the global optimum). The initial random search may not have helped as much in the 2-outputs-1-constraint task from Table 5 because one of the design variables was monotonic (i.e., an improvement in the objective function via detachment rate also moved it away from all constraints; the only other task with this feature was the 2-outputs-2-constraints task from Table 2), meaning it would benefit less from the broad

early search since the design task effectively only has three design variables of interest.

7.3 Validation of synergistic human-agent approach

One of the greatest benefits of the human-agent approach is the effectiveness it provides in improving human design processes without extensive cognitive studies, which are costly and time-consuming. Many users would have to be recruited to test a variety of strategies, because there is a limit on how many tasks one user can solve over the course of a cognitive study, in addition to the limit on the number of strategies they could utilize. Unlike agents, a user cannot repeat the same task over and over, or under different conditions, and remain unbiased. Such limitations are highlighted with regards to the experimental design of the final cognitive study. Here, users first solved three tasks with the Univariate Learning strategy and the second three with the Expert Application strategy, whereas agents could solve all six tasks with all strategies (Sect. 5). However, agent studies cannot run independently of cognitive studies if the end goal is to provide human engineers better search strategies, because the agents must find strategies that make sense to humans that eventually carry out the strategy.

The synergistic human-agent approach is generalizable to other domains beyond the myosin application. To summarize succinctly, it requires characterization of human search behaviors, automation via rule-based agent strategies that capture the cognitive basis of the human approaches, and then refinement of those approaches via rapid implementation with agents that returns a highly effective search approach to users. In this study, the agents implemented over one thousand approaches to determine which strategies are most effective and should be returned to users. With these new strategies in hand, the users performed much better than those with no provided strategy.

However, the best strategies found here may not be best for other design tasks of other domains, or even these design tasks with a different number of evaluations allowed. To elaborate, if a user was provided fewer than ten evaluations, the expert application approaches would be more useful because of their fast convergence, whereas methods with more random evaluations may be more useful as problem complexity grows with more variables. Therefore, it is our approach, not the specific strategy outcomes that will be most generalizable to other domains and tasks. We recommend deriving new strategies through human-agent approaches through careful consideration of the complexity unique to that domain's design space and the availability of gaining information concerning evaluated design performance. However, the findings from this study could inform which cognitive-based approaches to initially investigate in other domains.

Future implementation of the approach could consider methods of statistical data mining for finding trends in large sets of human data and more flexibility in the variances that agents are allowed in implementing a strategy. It is possible that other useful search strategies exist that could further improve user searches, such as those correlated with the amount of time a user has spent between design inputs which are indicative of analytical reasoning (long pauses between design evaluations) or heuristic reasoning (when a user quickly applies the same strategy repeatedly). The strategy produced by this study is representative of a univariate search approach that is known to be useful in human scientific reasoning (Kuhn et al. 2008) and is reflective of the cognitive preferences many users carried into their search approaches.

Univariate search behaviors, in addition to being familiar to users, may also be beneficial because they minimize cognitive load (Hirschi and Frey 2002) by reducing the number of variables and interactions a user has to remember and consider throughout their search. This notion is supported by our findings in the pilot and baseline cognitive studies that demonstrated users performed better when they made small changes to their design. Small changes to designs may not be the most effective strategy for all domains; from the current results we can only conclude that they are the most effective strategy found with consideration to the examined myosin design space for these particular tasks. Cognitive load associated with both the strategy a user chooses, and the specifics of their interface, could be further explored through experiments that measure cognitive load associated with the problem-solving process (Hart and Staveland 1988). More advanced strategies to consider in future work might involve: (1) the ways in which agents and users infer relative sensitivities among variables, (2) the most effective learning periods throughout a search endeavor and (3) interactions among variables (how one variable affects the gain of another in the design space).

8 Conclusion

This study has demonstrated the merit in developing strategies for designing complex systems through a synergistic human-agent approach. The methodology is implemented in the myosin domain to find highly effective cognitive-based search strategies, but is also generalizable to other domains and broadly to engineering decision-making processes. Our approach first required proposing possible cognitive-based search approaches, and then we developed a myosin GUI environment for measuring and assessing user search data in a pilot study. We then conducted a cognitive study with many users searching design

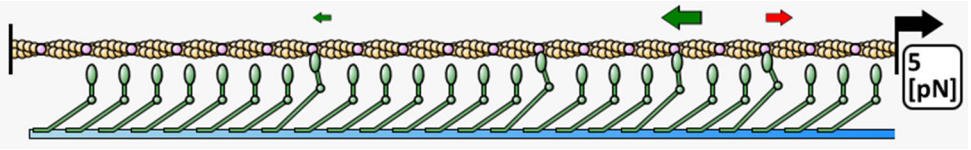

Fig. 8 Rendering of myosins interacting in a virtual environment (video available at http://youtu.be/kLX-0Zdizk8)

spaces and identified successful search trends based on the proposed cognitive search approaches. These trends were used to devise strategies that were rapidly implemented by software agents. Agents then solved tasks with varied strategic preferences until the most effective strategies were found and returned to users. It was found that when users implemented these agent-refined strategies, their search performance improved in comparison to users that had no imposed strategy.

This synergistic human-agent approach is particularly important when considering that human designers often have great difficulty in reasoning through the parameter interactions of complex systems. In our approach, agent-refined strategies were directly linked to how users naturally searched the problem. For instance, users that had the most success tended to make small changes to their best design and change one design variable at a time through univariate approaches. This finding, that users are successful when using univariate approaches, is congruent with univariate searches being a crucial pattern of reasoning in learning and applying knowledge in science-based tasks. The best strategies used by the agents, and returned to the humans, required consistent use of univariate searches to methodologically learn and apply knowledge via greedy search tactics. In some cases, a few random design evaluations early in the search process improved the likelihood of the greedy search converging on higher-quality designs. Most importantly, these univariate search tactics are robust for effectively solving many types of design tasks and facilitate both novice and domain expert approaches to search spaces.

The process of utilizing cognitive studies to isolate trends for strategies that are refined with software agents is generalizable to many other domains and is particularly useful as the agents can automate many different search strategies in comparison with time-consuming cognitive studies. Such a generalizable methodology is important, because it enables the discovery of effective strategies that are well suited for a particular application, which may include strategies that perform poorly in the myosin domain but are highly effective in other domains. In sum, our synergistic cognitive agent-based methodology resulted in highly effective search strategies for users, and future applications of the approach could aid in furthering engineers' capacities to design high-performance technologies of ever-increasing complexity.

Acknowledgments Partial funding for this study was provided by the National Defense Science and Engineering Graduate Fellowship and by the National Science Foundation under Grant CMMI-1160840. Patra Virasathienpornkul and Kirstie Pomilio Egan helped administer cognitive studies. Early iterations of this work were accepted to the 2013 International Conference on Engineering Design (Egan et al. 2013b) and the 2014 ASME International Design Theory and Methodology Conference (Egan et al. 2014).

Appendix

The appendix contains information for ("Myosin mathematical modeling") myosin mathematical modeling, ("Design task generation and difficulty") how design tasks are created and balanced, ("Pilot cognitive study with real-time and delayed feedback") pilot cognitive study results, ("Summary of best agent strategies") assessment of agents and all of their considered preferences, and ("Supplemental media") supplementary media.

Myosin mathematical modeling

A full account of myosin design is beyond this paper's scope, so only equations necessary for extending past myosin models for GUI implementation are presented. These equations borrow heavily from a previous well-established mathematical model (Howard 2001). Myosins operate through converting ATP molecules into mechanical energy and then stochastically attach and apply a force to an actin filament. Myosins have three structural states: attached and generating positive force, attached and generating negative force, or detached and generating no force. A myosin's performance is tunable via engineering its molecular configuration for altered mechanics (Anson et al. 1996) and chemistry (Murphy and Spudich 1998). For the GUI, three myosin design parameters are considered: (1) a myosin's lever arm length l, (2) a myosin's attachment rate k_{on}, and (3) a myosin's detachment rate k_{off}. Additionally, the number of myosins N_{myo} interacting with the filament is also considered as a system input parameter (Fig. 8).

Myosins are assumed to behave as linearly elastic elements once attached, utilize the energy of one ATP per cycle ε_{atp}, have an attachment rate dependent on the actin filament velocity v, and detach based on a time constant k_{off}^{-1} during their negatively strained state. Myosins are assumed to utilize the same energy per cycle ε_{atp} (62.5 zJ)

Table 6 Six NLP design tasks with varied numbers of outputs and constraints for users to solve during the pilot cognitive study

Design tasks for real-time/delayed feedback cognitive study				
Design goal	Goal constraint	Secondary constraint	Valid designs (%)	Task type
Highest system energy use	≤2.3 (ATP/ms)	None	~50	1-Output-1-constraint
Lowest avg # of attached myos	≥5.7 (N_{att})	None	~50	1-Output-1-constraint
Highest filament velocity	None	System energy use ≤2.3 (ATP/ms)	~50	2-Outputs-1-constraint
Lowest system energy use	None	Avg # of attached myos ≥5.0 (N_{att})	~50	2-Outputs-1-constraint
Lowest filament velocity	≤5.0 (μm/s)	System size ≥4.0 (nm^{-1})	~25	2-Outputs-2-constraints
Highest avg # of attached myos	≥5.4 (N_{att})	Filament velocity ≤3.2 (μm/s)	~25	2-Outputs-2-constraints

with the same efficiency regardless of their configuration. When exerting positive force, myosin lever arms of length l rotate θ (30°) relative to their head a linear distance $\delta_+ = l \cdot \sin(\theta)$. The stiffness κ of a myosin is therefore constrained to $\kappa = \varepsilon_{atp}/\delta_+^2$. The force generation of a myosin is dependent on its proportion of time attached to a filament r, which is found by dividing the distance a myosin remains attached to a filament $\delta_{on} = \delta_+ + \frac{v}{k_{off}}$ by the total distance the filament travels relative to a myosin during each myosin cycle Δ_c, provided by

$$\Delta_c = \frac{x_d}{1 - \exp\left(\frac{-k_{on} \cdot x_z}{v}\right)}. \quad (1)$$

x_d (36 nm) represents the spacing of myosin binding sites on actin, and x_z (1 nm) represents how close a myosin must be to a binding site to attach. The myosin duty ratio is $r = \delta_{on}/\Delta_c$. The time-average force $\langle f_{myo}(v) \rangle$ of an individual myosin is found through considering its time-average strain $\langle x_e(v) \rangle$ while attached, given by

$$\langle x_e(v) \rangle = \frac{\delta_+^2 - 2 \cdot (\delta_-(v))^2}{2 \cdot \delta_{on}(v)}, \quad (2)$$

and multiplied by a myosin's stiffness and duty ratio, expressed as

$$\langle f_{myo}(v) \rangle = \kappa \cdot r \cdot x_e(v). \quad (3)$$

The force of a system of myosins is $F_{sys} = N_{myo} \cdot \langle f_{myo} \rangle$, where N_{myo} represents the number of cycling myosins within reach of a single filament. It is not possible to solve for velocity as a function of force; therefore, a system response is calculated for display in the GUI by iteratively solving Eq. (3) with force as the independent variable. We assume a system force of 10 pN for all design tasks in the GUI. The average number of myosins attached to actin, $N_{att} = N_{myo} \cdot r$, is a stability metric of the system. It is suggested that at least two or more myosins must remain attached on average in order to maintain constant directionality of the actin filament and avoid system dissociation (Uyeda et al. 1990). The amount of ATP a system utilizes, E_{sys}, is determined by first calculating the average ATP used over time by each myosin, given by $e_{myo} = v/\Delta_c$, and then multiplying it by the total number of myosins, to find $E_{sys} = N_{myo} \cdot e_{myo}$. It is not necessary to calculate the precise area of the system for the purposes of this study, and a relative system size metric, A_{sys}, is found as a proportion between the number of myosins and myosin lever arm length $A = N_{myo}/l$ when calculated for GUI output.

Design task generation and difficulty

Six NLP formulations were developed to represent myosin design tasks for the pilot cognitive study (Sects. 3.3 and "Pilot cognitive study with real-time and delayed feedback") and are separated into three categories depending on the number of output parameters and constraints considered (Table 6). Each task had a design goal with respect to one output parameter as an objective function and then constraints either on the goal output or a secondary output. Possible objective functions and secondary outputs were the velocity of the actin filament propelled by myosins v, the rate of ATP consumption of all myosins in a system E_{sys} (i.e., energy usage), the average number of myosins attached to the filament N_{att} (a metric indicative of stability), and the system size A_{sys}. A user is expected to search the design space for a configuration that best fulfills a design goal without violating any constraints. Designs are all scored qualitatively on a scale of one to zero relative to the global objective function. If any constraint was violated the design was considered infeasible and given a score of zero. Each constraint added to the task effectively cuts the feasible design space by 50 %, which was implemented to maintain consistency in design task difficulty. For two constraint tasks, the secondary constraint is the only active constraint limiting global optimality. A variety of different combinations of goal and constraint outputs were chosen in order to reduce user learning and familiarity with the tasks.

To ensure that the design spaces represented problems of roughly the same difficulty for each pairing of design tasks, all 625 designs for a task were evaluated and graded relative to the global optimum solution for the task under consideration. Figure 9a displays the distribution of design

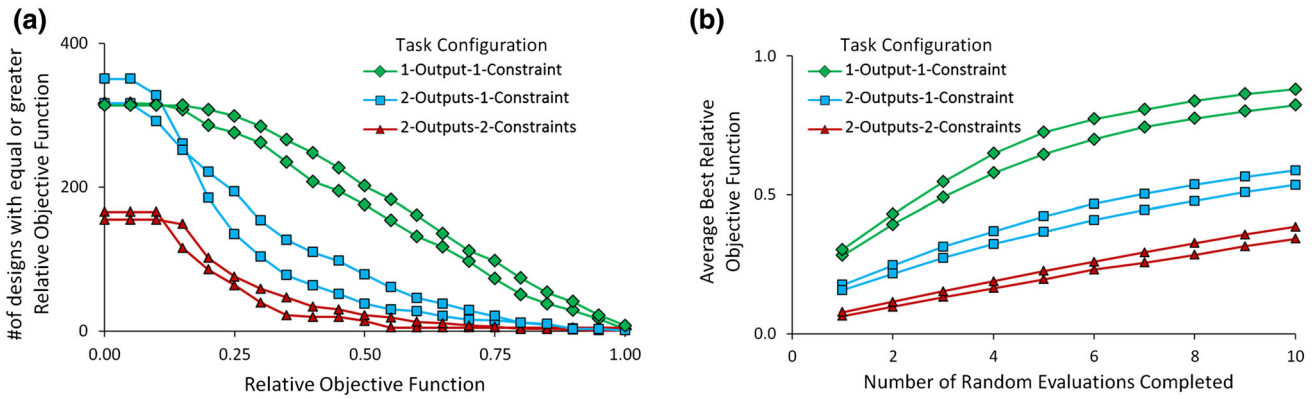

Fig. 9 **a** The number of designs within each scoring percentile relative to the global optimum and **b** mean best relative objective function when each task is solved with a random solver as a function of how many evaluations had been completed during a particular search

performance for tasks found in Table 6, where the total number of designs that have a given relative objective function value or higher is reported for each relative objective function value. As expected, more highly constrained designs have fewer feasible designs, and less designs near the optimal solution. More critically, the pairs of problems were very close in difficulty, if difficulty is assumed to correspond with the proportion of good designs near the optimal solution. A second assessment of problem task difficulty was conducted by comparing the results of a random solver for each task in Fig. 9b, which further supported that tasks with similar numbers of outputs and constraints result in comparable design spaces. When the random solver was used to solve each task, it evaluated each task with random inputs until its average performance could be found with negligible error after solving the tasks thousands of times.

Pilot cognitive study with real-time and delayed feedback

The pilot user study consisted of four participant groupings that each received a unique sequence of tasks to form a within-subject experiment that investigates the effects of feedback and task complexity on solution quality. In all tasks, users evaluated ten designs. In the real-time feedback condition, output information via the GUI was provided immediately to the user. With delayed feedback, users did not receive evaluated design performance information from the GUI until after their eighth evaluation. There were four groupings of users developed to form a within-subject experiment that minimized effects of learning across conditions and measured the effects of real-time versus delayed feedback for each task configuration. The first two groupings began with the first task in Table 6 and alternated tasks between the real-time and delayed feedback condition, with differing initial feedback conditions (i.e., the first group did tasks in Table 6 from top to bottom, beginning with real-time feedback, the second group also did Table 6 tasks from top to bottom, but began with delayed feedback. The feedback type was switched for every problem sequentially for both groups). The third and fourth groupings reversed the ordering of tasks from these first two groups (i.e., doing Table 6 tasks from bottom to top). For each task, a user was allowed ten evaluations and a maximum time of 3.5 min. Sixteen mechanical engineering students (4th and 5th year seniors, and graduate students) were recruited from Carnegie Mellon University and randomly and evenly assigned to the four study conditions.

All designs were graded relative to the global optimum for a given problem, by dividing the objective function found for that design by the global optimum to provide it a score between zero and one (the reciprocal was used for minimization tasks). For each task pair of a given difficulty, data was averaged and standard error was found for all real-time and delayed searches separately. A random solver was also used to solve each task, which acted as a control and repeatedly solved the tasks with random inputs until its average performance could be found with negligible error. Since real-time and delayed searches were varied between tasks for a user, each participant was represented once in both the real-time and delayed aggregations for a pair of tasks and the user was his/her own control for individual variability in human problem-solving performance. Results for each task and condition are plotted in Fig. 10a, with a user's best design being compared with the global optimum and given a relative score that is averaged with all other users in that same condition. In Fig. 10b, the same dependent measure is found, but users are separated according to whether they received the 1-output-1-constraint task initially or the 2-outputs-2-constraints task, rather than with respect to the study condition.

Users that received immediate feedback found designs with higher relative objective functions than those with

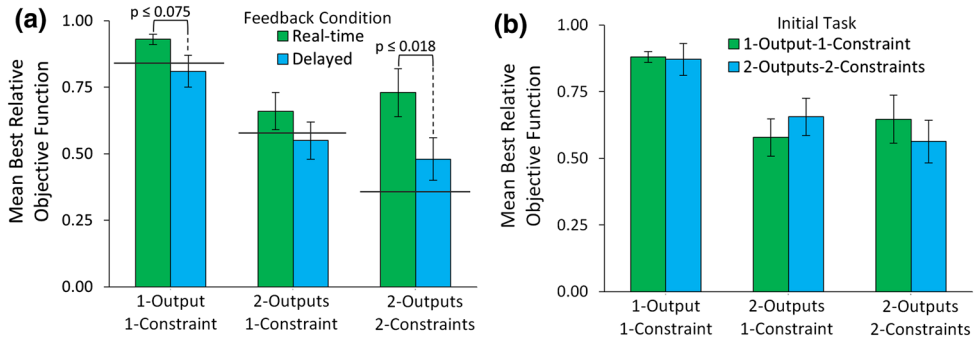

Fig. 10 Mean best relative objective function found by users under **a** real-time and delayed feedback conditions for each task type and **b** for both conditions aggregated based on the users' initial task type. All *error bars* represent standard error of the mean; significant ($p < 0.05$) and trending differences are indicated by reported *p*-values on charts. *Horizontal lines* indicate random solver results

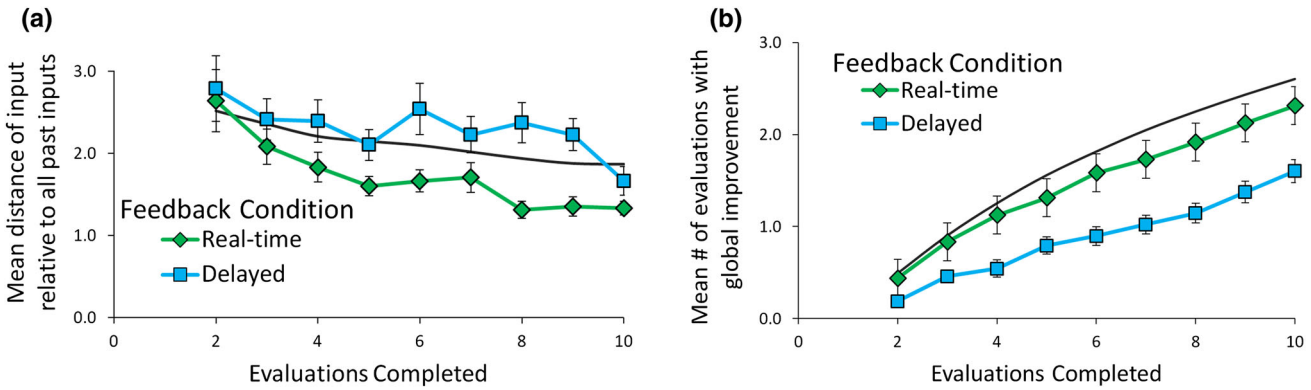

Fig. 11 Effects of feedback on user search behavior aggregated after each completed evaluation for **a** choosing design inputs near their previous best design and **b** how often they reach global improvement. All *error bars* represent standard error of the mean. *Dark lines* indicate random solver results

Table 7 Table detailing agent performance for each Tables 2 and 5 task, and the average of all tasks

Mean of all tasks	Baseline tasks (Table 2)			Additional tasks (Table 5)			Agent configuration			
	1-Out.-1-con.	2-Out.-1-con.	2-Out.-2-con.	1-Out.-1-con.	2-Out.-1-con.	2-Out.-2-con.	Initial guess	No. of random	Max dist from best	Rule-based agent strategy
0.84	0.95	0.54	0.91	0.91	0.82	0.93	Mid	0	4	Uni. Expert App.
0.84	0.94	0.62	0.90	0.89	0.80	0.91	Mid	1	4	Uni. Expert App.
0.82	0.93	0.66	0.86	0.87	0.78	0.86	Mid	3	4	Uni. Expert App.
0.79	0.94	0.49	0.82	0.91	0.74	0.85	Mid	0	3	Uni. Learn/Apply
0.78	0.93	0.53	0.80	0.88	0.73	0.82	Mid	1	3	Uni. Learn/Apply
0.77	0.92	0.63	0.68	0.86	0.73	0.81	Random	0	3	Uni. Learn/Apply
0.69	0.90	0.46	0.64	0.87	0.61	0.64	Mid	0	2	Uni. From Best
0.68	0.90	0.48	0.61	0.86	0.61	0.63	Mid	1	2	Uni. From Best
0.68	0.89	0.51	0.60	0.86	0.61	0.62	Mid	3	2	Uni. From Best
0.68	0.89	0.53	0.60	0.86	0.61	0.58	Mid	0	2	Stoch. From Best
0.67	0.86	0.57	0.58	0.82	0.60	0.57	Mid	0	3	Stoch. From Best
0.67	0.86	0.57	0.58	0.82	0.60	0.57	Mid	0	5	Stoch. From Best
0.67	0.84	0.63	0.53	0.82	0.61	0.55	Random	–	–	Rand. Solver

Results are sorted in descending order based on the value in the mean of all tasks column. Each row represents a different agent configuration

delayed output, which suggests that the output provided by the tool to users is helpful (Fig. 10a). The results also showed that condition differences were greatest on the most difficult task, and that users in the real-time condition were close to finding the global optimum on the first task as a group, which is indicative of reaching a ceiling of performance for the task. Figure 10b demonstrates that there was not significant learning across tasks. Therefore, future cognitive studies that are designed in a similar manner of varying tasks greatly among one another should also have minimal amounts of learning among users.

We further investigate whether real-time feedback aided users by investigating how search behavior and performance were affected during each point in the evaluation process. The mean distance of a user's design input compared with all past inputs was determined using the same method as described for the baseline study in Sect. 4.2 and results were aggregated according to user condition and tasks for each evaluated design during a search (Fig. 11a). Search performance was tracked throughout the solution process by counting how many times a user had globally improved their relative objective function at each point in the design process and is plotted in Fig. 11b for each condition.

Users in the delayed feedback condition made larger changes to their designs (Fig. 11a) over the course of their search than those in the real-time feedback condition, which is correlated with users in the delayed feedback condition also being less likely to improve their global best solution with each new evaluation (Fig. 11b). These findings suggest that users benefitted from real-time feedback by utilizing information to make smaller, more useful changes to their designs. However, users in the real-time condition still did not improve their design as often as a random solver (Fig. 11b), which suggests there is room for improving user search approaches.

Summary of best agent strategies

Agents with different preferences were used to solve tasks in over a thousand possible configurations by implementing one of four rule-based strategic approaches that include Univariate Expert Application, Univariate Learn and Apply, Univariate From Best, and Stochastic From Best as described in Sect. 5. Agent preference parameters included (1) an agent's initial design input preference (random, median of all inputs, all inputs set to their lowest, all inputs set to their highest), (2) how many random design evaluations follow their initial evaluation (0–8), and (3) how much maximum knowledge was obtainable/known a priori (only varied for knowledge-based strategies from zero to four).

Each agent solved each task 2500 times, and then the mean best relative objective function was found for each task, and all six tasks (Tables 2, 5) were aggregated for each agent. A partial summary of the best agent configuration is present in Table 7, where the best configurations were based on those with the highest mean best relative objective function value when averaged across all tasks. To add diversity to the table and show representation from each strategy, the global optimum agent configuration and then the two agent configurations that had an odd number of random evaluations and equivalent max distance from best parameters (with the exception of the Stochastic from best strategy since that was its only parameter that varied) are presented in Table 7. This small subset of solutions shows that increasing the number of random evaluations up to three in most cases does not greatly impede the robustness of a strategy, yet typically increases the design score of the 2-outputs-1-constraint task from Table 2. The table demonstrates tiers of strategy, with the best Univariate Expert Application strategies being universally better than the Univariate Learn/Apply strategies that tend to be universally better than Univariate From Best strategies.

Supplemental media

1. Myosin Description and Animation (http://youtu.be/kLX-0Zdizk8).
2. Software Tutorial and Practice Task (http://youtu.be/XOi3n-XwAXQ).
3. Myosin Table 2 Design Tasks with Generic Input–Output Names (http://youtu.be/-N–Jopb4DA).
4. Cognitive Study 3 Control Task, Strategy Tutorial, and Learning/Apply Task (http://www.youtube.com/watch?v=3psPSKYl9WE).

References

Aladahalli C, Cagan J, Shimada K (2007) Objective function effect based pattern search—theoretical framework inspired by 3D component layout. J Mech Des 129:243–254

Anson M, Geeves M, Kurzawa S, Manstein D (1996) Myosin motors with artificial lever arms. EMBO J 15:6069–6074

Bach A, Beier J, Stern-Staeter J, Horch R (2004) Skeletal muscle tissue engineering. J Cell Mol Med 8:413–422

Belegundu A, Chandrupatla T (2011) Optimization concepts and applications in engineering. Cambridge University Press, Cambridge

Buczak A, Cooper D, Hofmann M (2005) Evolutionary agent learning. Int J Gen Syst 35:213–254

Campbell M, Cagan J, Kotovsky K (1999) A-design: an agent-based approach to conceptual design in a dynamic environment. Res Eng Des 11:172–192

Chandra D, Bergmann F, Sauro H (2009) TinkerCell: modular CAD tool for synthetic biology. J Biol Eng 3:1–17

Chen Z, Klahr D (1999) All other things being equal: acquisition and transfer of the control of variables strategy. Child Dev 70:1098–1120

Chi M, Roscoe R, Slotta J, Roy M, Chase C (2012) Misconceived causal explanations for emergent processes. Cogn Sci 36:1–61
DuPont B, Cagan J (2012) An extended pattern search approach to wind farm layout optimization. J Mech Des 134:081002
Egan P, Cagan J, Schunn C, LeDuc P (2013a) Design of complex biologically based nanoscale systems using multi-agent simulations and structure–behavior–function representations. J Mech Des 135:061005
Egan P, Cagan J, Schunn C, LeDuc P (2013b) A modular design tool for visualizing complex multiscale systems. Paper presented at the international conference on engineering design, Seoul, South Korea
Egan P, Cagan J, Schunn C, LeDuc P (2014) Cognitive-based search strategies for complex bio- nanotechnology design derived through symbiotic human and agent-based approaches. Paper presented at the international conference on design theory and methodology, Buffalo, New York
Guckenheimer J, Ottino J (2008) Foundations for complex systems research in the physical sciences and engineering. Report from an NSF workshop
Hanna L, Cagan J (2009) Evolutionary multi-agent systems: an adaptive and dynamic approach to optimization. J Mech Des 131:011010-011011–011010-011018
Harada Y, Sakurada K, Aoki T, Thomas D, Yanagida T (1990) Mechanochemical coupling in actomyosin energy transduction studied by in vitro movement assay. J Mol Biol 216:49–68
Hart S, Staveland L (1988) Development of NASA-TLX (Task Load Index): results of empirical and theoretical research. Adv Psychol 52:139–183
Hirschi N, Frey D (2002) Cognition and complexity: an experiment on the effect of coupling in parameter design. Res Eng Des 13:123–131
Hmelo-Silver C, Marathe S, Liu L (2007) Fish swim, rocks sit, and lungs breathe: expert-novice understanding of complex systems. J Learn Sci 16:307–331
Howard J (2001) Mechanics of motor proteins and the cytoskeleton. Sinauer Associates, Sunderland
Kuhn D, Iordanou K, Pease M, Wirkala C (2008) Beyond control of variables: what needs to develop to achieve skilled scientific thinking? Cogn Dev 23:435–451
Landry L, Cagan J (2011) Protocol-based multi-agent systems: examining the effect of diversity, dynamism, and cooperation in heuristic optimization approaches ASME. J Mech Des 133:021001-021001–021001-021011
Michael S, Faeder J, Emonet T (2011) Efficient modeling, simulation and coarse-graining of biological complexity with NFsim. Nat Methods 8:177–185
Murphy C, Spudich J (1998) Dictyostelium myosin 25–50 K loop substitutions specifically affect ADP release rates. Biochemistry 37:6738–6744
Neiman V, Varghese S (2011) Synthetic bio-actuators and their applications in biomedicine. Smart Struct Syst 7:185–198
Nielsen J (2006) F-shaped pattern for reading web content Alertbox: current issues in web usability. Retrieved Oct 31, 2008 from http://www.useit.com/alertbox/reading_pattern.html
Olson J, Cagan J, Kotovsky K (2009) Unlocking organizational potential: a computational platform for investigating structural interdependence in design. J Mech Des 131:031001-031001–031001-031013
Ottino J (2004) Engineering complex systems. Nature 427:399
Pretz J (2008) Intuition versus analysis: strategy and experience in complex everyday problem solving. Mem Cogn 36:554–566
Schiaffino S, Amandi A (2009) Building an expert travel agent as a software agent. Expert Syst Appl 36:1291–1299
Shea K, Cagan J, Fenves S (1997) A shape annealing approach to optimal truss design with dynamic grouping of members. J Mech Des 119:388–394
Shrestha S, Lenz K (2007) Eye gaze patterns while searching vs. browsing a Website Usability News 9
Simpson T, Barron K, Rothrock L, Frecker M, Barton R, Ligetti C (2007a) Impact of response delay and training on user performance with text-based and graphical user interfaces for engineering design. Res Eng Des 18:49–65
Simpson T, Frecker M, Barton R, Rothrock L (2007b) Graphical and text-based design interfaces for parameter design of an I-beam, desk lamp, aircraft wing, and job shop manufacturing system. Eng Comput 23:93–107
Uyeda T, Kron S, Spudich J (1990) Myosin step size estimation from slow sliding movement of actin over low densities of heavy meromyosin. J Mol Biol 214:699–710
Vattam S, Goel A, Rugaber S, Hmelo-Silver C, Jordan R, Gray S, Sinha S (2011) Understanding complex natural systems by articulating structure–behavior–function models. Educ Technol Soc 14:66–81
Viciana-Abad R, Reyes-Lecuona A, Poyade M, Escolano J (2011) The role of mismatches in the sensory feedback provided to indicate selection within a virtual environment. Multimed Tools Appl 55:353–378
Villalobos A, Ness J, Gustafsson C, Minshull J, Govindarajan S (2006) Gene Designer: a synthetic biology tool for constructing artificial DNA segments. BMC Bioinform 7:1–8
Wolf D, Hyland J, Simpson T, Zhang X (2011) The importance of training for interactive trade space exploration: a study of novice and expert users. J Comput Inf Sci Eng 11:031009
Zhang X, Simpson T, Frecker M, Lesieutre G (2012) Supporting knowledge exploration and discovery in multi-dimensional data with interactive multiscale visualisation. J Eng Des 23:23–47