Incorporating Cognitive Principles into a Computational Engineering Design Methodology

Jarrod Moss
Carnegie Mellon University
Senior Honors Thesis

Advisors: Kenneth Kotovsky, Ph.D.
Jonathan Cagan, Ph.D.

Introduction

Engineering design is a complex domain in which a number of processes and people interact to produce the desired product. Throughout the design process engineers utilize a large body of personal experience and design knowledge in order to create a functioning device. It is almost impossible to imagine a skilled engineer attempting to solve a problem without relying on some form of previous domain knowledge. Exactly how they accumulate this knowledge and later decide to use it in a design problem is not known, but these processes can be explored using tools and methods from cognitive psychology.

The amount of domain knowledge and experience, as well as the organization of this information, in memory is one of the main differences between expert and novice problem solvers in domains such as chess and physics (Larkin, McDermott, Simon, & Simon, 1980; Richman, Staszewski, & Simon, 1995; Simon & Simon, 1989). If this result holds for engineering design as well, then understanding the processes that occur as engineers store and recall knowledge would have implications for design education. An understanding of the memory processes that go on in design would also allow the creation of better computational design aids as well as enhancing tools that automate parts of the design process. All of these benefits serve to increase the productivity of designers while generating better quality products.

Studying the domain of engineering design from a cognitive standpoint has the potential to produce an even greater number of benefits for cognitive psychology. First, there are a number of general cognitive architectures such as ACT-R (Anderson & Lebiere, 1998) and SOAR (Newell, 1990) as well as memory models, such as EPAM (Richman et al., 1995), that could be further evaluated by applying them to the domain of engineering design. An even more interesting area of cognition that could be examined within this domain is creativity. Creativity is a complex area of cognition that most models and theories of cognition have not adequately explained or in many cases even explored. Varying degrees of creativity appear regularly in engineering design, and therefore any successful cognitive model of engineering design would have to account for some of the creative processes that go into solving design problems.

The fields of cognitive psychology and design can both benefit from research that examines the cognitive processes that occur in design, but a complete cognitive model of engineering design is a far off goal. There are more immediate benefits that can be obtained by including some general cognitive principles into current design methodologies. There are a number of existing design models and methodologies that attempt to describe some of the cognitive processes that go into design. One such model is the TEA (task/episode accumulation) model proposed by Ullman, Dietterich, and Stauffer (1988), and another such model is the model proposed by Adelson (1989), which incorporates a basic model of how analogy might operate within the domain. Other researchers have focused on computational methods for the domain (Welch & Dixon, 1994; Campbell, Cagan, & Kotovsky, 1999, 2000). Some of these models were meant to help explain the design process, while others have attempted to automate parts of the design process or at least produce computational aids to assist the designer. In particular, the A-Design system automates the conceptual design process (Campbell et al., 1999, 2000).

This paper investigates the effects of incorporating some basic learning and memory processes into the A-Design methodology. These processes are inspired by what we know of human cognitive processes, and they are expected to significantly decrease the amount of time that it takes for A-Design to construct good designs. The effects of these cognitive-like processes are evaluated in a series of experiments, but first, a brief overview of the A-Design process is presented. This overview is followed by an explanation of the modifications that were made to incorporate the knowledge base and associated memory processes, and then some empirical results are discussed that demonstrate the effects of utilizing previous design knowledge on some general electromechanical design problems.

A-Design

A-Design is a multi-agent computational system that automates the conceptual design process, and it is currently capable of solving several electromechanical design problems. A brief overview of the structure and capabilities of the A-

Design system is necessary in order to present the results of this current research, however a more detailed description of the system can be found in (Campbell, Cagan, & Kotovsky, 1999).

A design problem is specified to A-Design by encoding the input and output constraints of the desired device into A-Design's representation. The example of a punch press device will be used throughout this description of the design generation process (see Figure 1). The input to the punch press is a downward force on a handle, and the output of the device is a much larger downward force that drives a punch into some material. Also, the punch should only traverse a distance of .25 m. Every design problem given to A-Design also specifies a number of objectives that the system should try to optimize. For example, in the punchpress problem the cost of the design should be kept to a minimum, the distance that the input lever is displaced should be minimized, and the design should come as close as possible to the given specifications for the output force and the distance the punch is displaced. The cost of the design can be evaluated by summing the cost of the individual components, but the other objectives are not as easy to evaluate. The distance that the input handle has to be displaced should be minimized, and in order to evaluate this objective, A-Design extracts behavioral equations from a completed design and evaluates how close the actual displacement comes to the ideal displacement of zero. The other two objectives measure how close a given design comes to the specifications of the design problem. For example, suppose that the punchpress lever will always be given 6 N of force, and an output force of 100 N is desired along with a punch displacement of .25 cm. Using these given specifications, A-Design will evaluate the output force and punch displacement of a given design by solving the appropriate behavioral equations with 6 N of force as the input. Being able to evaluate a design automatically is essential to the A-Design process because these evaluations allow the selection of current designs and the generation of future designs.

Figure 1. The specification for the punch press problem.

The adaptive selection and generation of designs is accomplished through an iterative design process (see Figure 2). In each iteration of the design process, the various software agents interact to produce an array of candidate designs that satisfy the given problem constraints. These designs are all evaluated on the multiple objectives specified for the problem, and the designs are separated into three groups: pareto, good, and poor. The designs that are selected from a design population actually consist of the Pareto group and the good group, while the poor group of designs is eliminated from consideration. The Pareto group is found by finding the Pareto optimal set of designs as they are evaluated on the multiple objectives, and the good set of designs is found by collapsing the multiple objectives of the design problem into a linearly weighted sum that is defined by the user of the system. Further details and definition of how these groups are determined can be found in Campbell et al. (1999). The Pareto designs automatically become part of the next iteration's population, and both the Pareto and good designs are allowed to "reproduce", in a manner described below, in the hope that some of their offspring will be better designs than the original. This adaptive design generation and selection process is similar to the concept of a genetic algorithm (Mitchell, 1996). The system will continue to develop and propagate designs to the next iteration until no further progress is apparent or a maximum number of iterations has been reached.

Figure 2. An iteration in A-Design is a complete cycle that starts with the C-agents.

The actual construction of a candidate solution to the design problem begins with the group of creator agents (C-agents). C-agents select embodiments from A-Design's embodiment library and add these embodiments to an incomplete design. Currently A-Design's embodiment library contains 25 different types of embodiments, which include components such as springs, gears, resistors, and motors. The actual embodiment selected from the library and the manner in which the embodiment is connected to the incomplete design depend on each C-agent's unique set of preferences. For example, a C-agent can prefer a specific physical domain, such as the electrical or hydraulic domains, and the C-agent could prefer to connect components in a parallel rather than serial fashion. As components are added to an incomplete design, the design's properties are updated until the design satisfies the given input and output constraints of the design problem or

until it is judged that the given design will probably never satisfy the design constraints. An example of the latter case is when all components in the design are connected to each other or to ground, and therefore there aren't any additional locations where another component can be added. After a set of candidate designs has been constructed, the designs are passed to the instantiation agents (I-agents) so that the conceptual design can be instantiated with real-world components. These agents select from a catalog of real components and instantiate a design by assigning these real components to the embodiments in a given design. For example, a design might contain a spring embodiment that is instantiated by an I-agent with a spring from the catalog of components, and this instantiated spring will therefore have specific values for attributes such as length, width, cost, and spring constant. This instantiation is a necessary process because it allows the behavioral equations to be extracted, which make it possible to evaluate the design's performance with regard to the design objectives.

The population of instantiated designs is passed to the manager agents (M-agents), and these agents guide the overall design process through collection of C/I-agent statistics and trend extraction. The M-agents keep track of the number of good designs that each C/I-agent contributes to, and the probability that a specific C/I-agent will be called upon in the future is a function of its past success. More importantly, the M-agents can extract current trends from the design population. A trend is a particular set of agents or components that is present in a set of designs, and trends are looked for in both the best and worst designs of a particular iteration. The trends from the best designs are passed back to the C/I-agents as a "todo" list, and the trends from the worst designs are passed back as a "taboo" list. Agents will try to avoid items on the taboo list while trying to produce items on the todo list. In this manner, the todo/taboo lists allow the M-agent to influence the design candidates that are generated in the next iteration of the design process.

In addition to keeping the good designs from the current iteration, all good and Pareto designs are passed to fragmentation agents (F-agents), who take out one or more components of the design. These fragmented designs are then reconstructed and become part of the next iteration's design population. This fragmentation process allows good designs to propagate similar designs to the next iteration with the hope that the changes made to the design will improve it. This process is the basis of the A-Design system, and using this method, the system can successfully solve a number of electromechanical design problems, some of which are described below.

Design Memory

Human designers benefit from the ability to learn from their experiences, and this knowledge in turn allows them to solve design problems more efficiently because key parts of a design can be recalled from memory. The ability to design an electromechanical devise requires knowledge of a number of things including such things as the types of components that are available to construct a device as well as how these components interact to achieve the desired functionality. One difference between experts and novices in a wide variety of domains is the size and organization of these chunks in memory (Richman et al., 1995). A-Design already had basic knowledge of the electromechanical domain, but the system lacked a way to store larger chunks of useful design information as well as a way to organize and retrieve these chunks from memory. The A-Design process might benefit from possessing such a learning capability just as expert designers benefit from the knowledge they have obtained through experience. In order to begin examining the beneficial effects that previous design experience might have on A-Design, the system was given the capability to store and retrieve useful design knowledge. Specifically, A-Design was given the capability to learn and remember good design chunks from design problems it has solved, and these chunks can later be used when solving new problems. A design chunk is a group of components that appear in a design and are connected to one another. A chunk could be a whole design, such as the pressure gauge discussed above, or it could be a subset of the components in a design. For example, the design chunk of a belt and pulley would include the two components, a belt and a pulley, as well as how these components are connected. Furthermore, a design chunk also contains input and output constraint information so that it can be used correctly in a later design.

The extraction of design chunks from a population of designs utilizes the existing trend detection functions already in place within A-Design. The system will automatically detect trends

that include subsystems of components, and it can transfer these trends to memory along with the information necessary for the subsystems to be reconstructed at a later time. The storage of such information brings up issues such as the structure and retrieval mechanisms of this memory system. These issues are important, and they ask questions that can be at least partially answered by research within cognitive psychology. Since human designers can store information regarding useful subsystems and can recall these subsystems for future use, then why not try to model A-Design's memory on this successful existing system? The actual structure and mechanisms of a designer's memory are not known exactly, but there are general memory models available within the field of cognitive psychology. One such memory model is incorporated in a general architecture of cognition called ACT-R (Anderson & Lebiere, 1998). ACT-R can successfully model a variety of human cognitive activities, including those that require memory encoding and retrieval. The ACT-R model of declarative memory was chosen because it is a well developed model, and the code for the system is freely available.

The ACT-R theory is a production system model of human cognition that incorporates a model of procedural and declarative memory. Declarative memory in ACT-R is a collection of chunks, and these chunks contain slots that identify values and properties associated with the chunk. For example, a design chunk generated by A-Design could look like the one in Figure 3. In this chunk the *isa* slot indicates that the chunk is a design chunk, the input domain to the chunk is translational, and the input interface is a bolt. This chunk is one example of a design subsystem that was learned from the punch press problem. This design chunk represents the subsystem formed by connecting together the hydraulic ports of two hydraulic cylinders. The connectivity information in Figure 3 is a translation of A-Design's internal representation of the connectivity information in a chunk. This particular chunk is useful in the punch press problem because two cylinders connected in this manner is a good way of amplifying the input force of the punch press.

Figure 3. An example design chunk learned by A-Design in the punch press problem.

Since ACT-R is a production system, chunks are retrieved from memory by matching to a production rule, and the exact matching is determined through a spreading activation model of memory that is discussed in Anderson and Lebiere (1998). Activation is spread based on the values of the slots in a chunk, which means that information can be retrieved from memory based on any of the slots of a chunk. In A-Design, this model means that a design chunk can be retrieved based on its input constraints, its output constraints, or both. A chunk could also be retrieved based on its connectivity information, but this retrieval function was not implemented because it seems unlikely that a human designer would retrieve design chunks from memory in this manner. A designer might retrieve information based on how components are connected, but it is not likely that this retrieval is as detailed as A-Design's current representation of connectivity information. Limiting A-Design's retrieval mechanisms to input and output information of the design chunk seems appropriate because a human designer could reasonably be expected to retrieve different subsystems from memory based on the input and output characteristics of the design he is focusing on. For example, a chunk representing the brake system of a car could be retrieved from memory when the current design problem calls for a subsystem that takes an input displacement and produces a decrease in velocity as output.

In the modified A-Design method, the system will automatically store design chunks based on the elements that appear on the todo list in the final iteration of a design problem. So when the system has finished solving a problem, it will record the subsystems that appear in the good designs. A-Design can retrieve these design chunks in a later problem using one of three different strategies, and three new C-agents (called Mem-agents, see Figure 2) were added to A-Design to embody these strategies. As the C-agents are constructing designs, if one of the Mem-agents is called upon to add a

component to an incomplete design, that agent retrieves a suitable chunk from memory and adds it to the design. The agents focus on a particular part of an incomplete design, and they try to retrieve a design chunk from memory based on the input/output constraints of this part of the design. As mentioned, there are three different Mem-agents, and they retrieve chunks from memory based on different properties of the incomplete design. One agent retrieves chunks from memory that match the current input constraints of the incomplete design, another agent focuses on the output constraints of the design, and a third agent tries to retrieve chunks that match both the input and output constraints. Only the input constraint Mem-agent is currently implemented in the system; the other agents will be completed at a later time, which is a relatively small step. These new agents along with the memory store are the core of A-Design's new learning and memory capabilities.

Design Problems

There are three mechanical design problems that are used in the following experiments: a punch press, a pressure gauge, and a weighing machine. The punch press problem is the same as described above, a handle is pulled which forces a punch through some material at the output. Punch presses are evaluated based on their cost, the amount of input handle displacement, and how closely they conform to the specified output displacement and force. The pressure gauge problem has an input pressure source and the output is a dial display that reflects the amount of pressure coming from the pressure source, and this problem is evaluated on the cost, mass, efficiency, and dial accuracy of the gauge. The weighing machine takes a force input on a footpad and has a dial output, and it is evaluated on the cost, mass, dial accuracy, and input displacement of the device.

It was necessary to add these problems to the set of problems currently solvable by A-Design so that the design memory addition could be thoroughly tested, and so I implemented them in A-Design's input/output constraint specification. I also made modifications to the embodiment library since most of the embodiments in the current library were originally included for the weighing machine problem. These specific problems were added to provide both a problem that was similar to the weighing machine as well as a problem that was significantly different from the weighing machine. The pressure gauge is a measurement device with a dial output just like the weighing machine, but the goal of the punch press is to amplify the small input force so that it is sufficient to drive a punch through some material. This problem shares little with the weighing machine, and it was added to evaluate how context dependent A-Design's new knowledge capabilities would be. For example, if A-Design learned design chunks from the weighing machine problem, then these chunks might be easier to apply in the pressure gauge problem since this problem is similar to the weighing machine. On the other hand, applying these same design chunks to the punch press might be more difficult and potentially less useful because the punch press does not have much in common with the weighing machine.

Experiments

Method

Two experiments were run to determine if the added memory element would benefit the A-Design process. The first experiment evaluated the effects of design memory on the pressure gauge problem. A-Design first solved the problem without the new memory, and then the system solved the weighing machine problem and stored design chunks from successful weighing machines. Next, A-Design again attempted the pressure gauge problem, but this time the system had knowledge of the weighing machine design chunks. This procedure was repeated for the punch press problem in place of the pressure gauge problem. In both experiments, the solution time and quality of the final design produced for the problem solved without memory were compared to the solution time and quality of the final design produced for the problem with knowledge of the weighing machine problem.

On each problem, A-Design was run 20 times, and each of one of these runs will be referred to as a trial. A total of 20 different design populations were produced for each problem, one on each trial. Each of these 20 trials was run for a total of 60 iterations of the design process, and each trial had a design population of size 130. The same weighing machine

runs were used for both of the experiments. For example, the chunks learned from the first trial of the weighing machine problem were used in both the first trial of the pressure gauge problem and the first trial of the punch press problem, the chunks learned from the second trial of the weighing machine problem were used in both the second trial of the pressure gauge problem and the second trial of the punch press problem, and so on. Each design problem had a user evaluation function defined for it. This function allows designs evaluated on a number of objectives to be collapsed to one value so that they can be compared. This user evaluation function is currently just a linear weighted sum of the values for the different objectives, and since all the objectives are minimization objectives, lower evaluation scores mean better designs. In each of the design problems, the weights were defined to emphasize the objective that seemed the most difficult for A-Design to optimize. The weights were chosen in this manner because it seemed most interesting to let A-Design focus on the most difficult design objective for each problem. For the weighing machine and pressure gauge, dial accuracy was emphasized, and for the punch press, the output force objective was emphasized.

Results

Pressure Gauge

For each iteration of the problem without memory, the user evaluation for the best design in each of the 20 trials was averaged. This average user evaluation is plotted in Figure 4 for each of the 60 iterations in the design problem. The same procedure was used to generate average user evaluations for the 60 iterations of the pressure gauge problem with memory. The curve for the trials with memory appears to be lower on average than the curve for trials without memory. To determine if the availability of memory allowed A-Design to produce better designs, the user evaluations of the best design for each of the 20 runs in the problem with memory were compared to the corresponding evaluations for the problem without memory. The mean for the without memory condition was 32203, and the mean for the with memory condition was 26607, sd1 = 6296, sd2 = 3083. A t-test was performed between these two sets of evaluations, and the results were significant, $t(27) = 3.57$, $p = .0014$. The addition of memory to A-Design allowed the system to produce better quality designs. Another way of assessing the benefits of design memory is to determine the amount of time saved by solving a problem with memory. To determine if the A-Design process with memory produced designs equivalent to those of A-Design without memory in a smaller number of iterations, a cutoff value was established by taking the average evaluation of the best designs produced by A-Design without memory for the design problem in question (the value for iteration 60 for the without memory condition in Figure 4). The number of iterations that A-Design with memory required to reach or surpass this cutoff value was compared to corresponding number of iterations for A-Design without memory (if a trial never surpassed this cutoff, then the value of 60 iterations was recorded for that trial). The mean number of iterations required in the without memory condition was 36.6 as compared to 21.4 for the with memory condition, sd1 = 21.2 and sd2 = 14.2. Another t-test was run for these two sets of numbers, and the results were S1, $t(33) = 2.67$, $p = .012$. The memory condition reduces the number of iterations necessary to reach the cutoff value by about 50 percent.

Punch Press

The same types of analyses were also conducted for the punch press problem. The results of the experiment can be seen in Figure 5. This graph gives the impression that having memory of the weighing machine problem may actually impede progress on the punch press problem. However, this observation may not be completely correct, because there appear to be two exceptionally bad trials included in the with memory condition. In both of these trials, A-Design failed to make progress on the problem until the very end of the set of 60 iterations. Figure 6 shows the means from the punch press problem if these two trials are not included in the dataset. In this graph, it appears that memory of the weighing machine did help in designing a punch press. The mean evaluation of the best designs for the without memory condition was 281,164 as compared to 250,991for the with memory condition, and both conditions had a standard deviation of around 580,000. The t-test for these groups was insignificant, $t = 0.16$, $p = 0.87$. A cutoff value was determined by using

the same method described above, and the mean number of iterations required to reach the cutoff for the without memory condition was 28.6 as compared to 25.4 for the with memory condition, and both conditions had a standard deviation of around 20. The t-test for these groups revealed insignificant differences, $t = 0.53$, $p = 0.60$. Overall, even though the graph of Figure 6 seems to indicate that memory of the weighing machine problem improved A-Design's performance on the punch press problem, these differences are not statistically significant.

[pic]**Figure 4. The average best design over all 20 trials for both memory conditions of the pressure gauge.**

[pic]Figure 5. The average best design over all 20 trials for both memory conditions of the punch press.

[pic]**Figure 6. The average best design over all 20 trials for both memory conditions for the** punch press excluding two trials in the with memory condition.

Discussion

A-Design does benefit from the capability to remember and later utilize useful design subsystems, and the system displays the hypothesized result that transfer of knowledge occurs to a greater extent when the problems are similar. The results indicate that A-Design produced better designs with memory at least in the pressure gauge problem. With previous design knowledge, it was able to produce designs equivalent to the best designs generated without memory in about half the number of iterations, again only on the pressure gauge problem. On the other hand, there were no significant improvements for the punch press problem. The reason for the difference in results for the two problems is probably due to the similarity (or lack thereof) between the weighing machine problem and the two problems tested (pressure gauge and punch press). The weighing machine problem and the pressure gauge problem are very similar both in their function and output. The function of both devices is measurement, and both devices display the measurement on a dial interface. A weighing machine might therefore have many of the same components as a pressure gauge, and these components will probably connect in similar ways. This overlap means that chunks learned by solving the weighing machine problem are very likely to be useful when designing a pressure gauge. On the other hand, the weighing machine and the punch press problems differ in a number of ways. The main function of the punch press is to amplify an input force, which doesn't have any readily observable mapping on to the measurement function of the weighing machine. Also the input and output interfaces between these two devices differ significantly.

These results might imply that previous knowledge only helps if the knowledge was learned on a problem similar to that on which it is being applied. This result would be consistent with findings related to transfer of knowledge in human problem solving. Holyoak (1990) demonstrated that when the domains or contexts of two problems were similar, subjects were more likely to transfer knowledge across problems than they were when the problems differed. Transfer of knowledge across problem boundaries can occur by a number of both explicit and implicit methods. One of the well known explicit processes for transfer is analogy, and the domain of engineering design seems to be a domain in which analogy is a common process. While the process by which A-Design remembers and retrieves design chunks is far from an explicit model of analogy in engineering design, there is reason to hope that success with an implicit mode of transfer like that which now occurs in A-Design could lead to the beginnings of a general model of analogy in the design domain.

Another unexpected result that seems to be apparent in the data is that todo learning, the todo list, significantly aids the design process when memory agents are present in the system. In all of the preceding experiments, todo learning was turned on during the 25th iteration of the design process. At this point in time, the lines for the with and without memory conditions really begin to diverge in Figure 4. Additionally, memory agents usually only contribute directly (adding an embodiment) to a design in about 2-3 of the top 10 designs in each trial of the pressure gauge problem. These results seem to indicate that some other process might be responsible for the divergence of the two lines in Figure 4. Also, in the punch press graph, there seems to be a significant improvement soon after todo learning is turned on. I hypothesize that the Mem-agents do contribute to a few designs throughout the design process, but when todo learning is turned on the system is able to propagate the good actions of the Mem-agents throughout the design population. For example if a Mem-agent has made an exceptionally good contribution to one design, then it is likely that this design will become part of the pareto or good groups, and so this design will be kept in the design population from one iteration to another. When todo

learning is turned on, the system extracts design subsystems from the best designs. The design that the Mem-agent successfully contributed to will likely be one of these best designs, and at least part of the subsystem that the Mem-agent contributed might be included on the todo list for subsequent iterations. The Mem-agents therefore make a further indirect contribution to the design process because their contributions will likely be propagated to other designs through their inclusion on the todo list. This hypothesis needs to be tested further, but it is an unexpected and surprising finding that results from the successful interaction of two independently developed components of the A-Design system.

If the addition of a simple form of design memory to A-Design can generate significantly improved results such as these, what might other more substantial cognitive processes contribute to the design process? The results from this study point to questions about how human designers learn from their design experiences and how they use their experience and domain knowledge in future problems. Another interesting phenomenon in design is the creative leaps that designers make in solving design problems. What makes these creative leaps possible and how are they different from routine design? In order to answer these and other questions, it is necessary to begin a more complete analysis of the domain. As mentioned in the introduction, some researchers have produced models of the design process, but none of these provides a satisfying explanation of the cognitive processes that underlie this highly complex domain. Research developing a cognitive model of design stands to benefit the domain of engineering design as well as psychology. These benefits could include improvements in design education, better computational design aids, design automation tools, an exploration of the creative processes occurring in design, and further expansion and validation of current cognitive architectures. A good deal of work remains to be done, but current models and methods from cognitive psychology can be utilized to facilitate the task.

References

Adelson, B. (1989). Cognitive research: Uncovering how designers design; Cognitive modeling: Explaining and predicting how designers design. *Research in Engineering Design, 1,* 35-42.

Anderson, J. R., & Lebierre, C. (1998). *The atomic components of thought.* Mahwah, NJ: Lawrence Erlbaum Associates.

Chase, W. G., & Simon H. A. (1973). Perception in chess. *Cognitive Psychology, 4*(1), 55-81.

Campbell, M. I., Cagan, J., & Kotovsky, K. (1999). A-design: An agent-based approach to conceptual design in a dynamic environment. *Research in Engineering Design, 11*, 172-192.

Campbell, M. I., Cagan, J., & Kotovsky, K. (2000). Agent-based synthesis of electromechanical design configurations. *Journal of Mechanical Design, 122,* 61-69.

Holyoak, K. J. (1990). Problem solving. In D. N. Osherson & E. E. Smith (Eds.), *An invitation to cognitive science: Vol. 3. Thinking* (pp. 116-146). Cambridge, MA: MIT Press.

Larkin, J. H., McDermott, J., Simon, D. P., & Simon, H. A. (1980). Expert and novice performance in solving physics problems. *Science, 208*, 1335-1342.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms.* Cambridge, MA: MIT Press.

Newell, A. (1990). *Unified theories of cognition.* Cambridge, MA: Harvard University Press.

Richman, H. B., Staszewski, J. J., & Simon, H. A. (1995). Simulation of expert memory using EPAM IV. *Psychological Review, 102*(2), 305-330.

Ullman, D. G., Dietterich, T. G., & Stauffer, L. A. (1988). A model of the mechanical design process based on empirical data. *AI EDAM, 2*(1), 33-52.

Welch, R. V. & Dixon, J. R. (1994). Guiding conceptual design through behavioral reasoning. *Research in Engineering Design, 6,* 169-188.

Acknowledgements

Thanks to Ken Kotovsky and Jon Cagan for advising me throughout this project, Matt Campbell for help with the details of A-Design, and Laura Gonnerman for ideas related to the analysis of my results.